译者序

数据挖掘这一学科近年来发展十分迅速，不仅产生了大量不同类型的挖掘算法，而且也表现出与机器学习等学科深度融合的态势。无论是从事研究的专家学者还是从事应用的开发人员都十分希望能一窥其大略，从而比较准确地把握数据挖掘领域当前的主干技术，并比较全面地了解当前的发展趋势。

当前，在市场上流通的数据挖掘方面的著作已经不算少了，主要是两大类：一类是具有完整体系的教材类图书，一类是面向特定领域的应用型图书。前者主要是服务教学，所以侧重原理、逻辑严谨，但是通常对数据挖掘的前沿介绍比较欠缺。后者往往集中于介绍某一领域的问题和方法，或者是关于某些典型工具的使用方法，其优点在于直观有效，但相对于整个数据挖掘领域其覆盖面偏小。

为此，很有必要对整个数据挖掘领域的近期发展和前沿成果进行梳理，而这一类信息往往散见于相关的大量学术期刊和会议文集中，限于视野和精力，任何个人都难以完成这一任务。在此基础上，还需要对当前庞大的数据挖掘知识体系进行恰当的取舍和凝练，这一工作必须依靠该领域的高水平学者。所以，国际数据挖掘社区合众人之力，在 2006 年推出了 *The Top Ten Algorithms in Data Mining* 这一继往开来之作。该书列举了评选出来的十个最具影响力的数据挖掘算法：C4.5、k-means、SVM、Apriori、EM、PageRank、AdaBoost、kNN、Naive Bayes 和 CART。我们认为该书有其鲜明特色：

第一，立意承前启后，推出的时机恰当。该书的内容涵盖了分类、聚类、统计学习、关联分析和链接分析等重要主题在近年来的发展，这不但对数据挖掘的研究和发展十分重要，也将数据挖掘推动到更大范围的真实应用中，激励更多数据挖掘领域的学者对这些算法的作用和新问题进行深入探索。

第二，汇集群体智慧，具有很高权威性。参评人员囊括了历届 ACM KDD 创新奖和 IEEE ICDM 研究贡献奖得主这些顶尖学者，以及 SIGKDD、ICDM 和 SDM 这三大数据挖掘学术会议的程序委员会的全体委员。此外，还组织了专题会邀请了一百多位领域专家进行开放研讨。

第三，执行过程严谨，确保内容高品质。第一阶段是由顶尖学者推荐算法并提供算法名称、简要理由和代表文献这些必要信息，第二阶段用 google scholar 对每个提名算法进行客观地引用验证和排序，第三个阶段由数据挖掘社区的专家和相关领域的专家进行投票，获得完全一致的结果。最后，邀请资深学者撰写上榜算法的介绍并集结成书。

本书的翻译工作由中科院软件研究所李文波和北京市科学技术情报研究所吴素研共同完成，我们非常希望能为国内数据挖掘方面的工作略尽一点绵薄之力，但是由于水平有限，译作难免有错漏之处，请不吝指出。

最后，我们还要感谢国家自然科学基金（项目编号：61003117）、ISTIC-Thomson Reuters 科学计量学联合实验室开放基金（项目编号：IT2011003）、国家 863 计划（项目编号：2013AA01A603）和国家软科学计划（项目编号：2009GXQ6D154）对我们工作的支持。

前　言

在香港举办的2006年度IEEE数据挖掘国际会议(ICDM, http://www.cs.uvm.edu/~icdm/)上,与会专家遴选出了十个最具影响力的数据挖掘算法,也就是本书所列的十个算法:C4.5、k-means、SVM、Apriori、EM、PageRank、AdaBoost、kNN、Naive Bayes和CART。

遴选过程第一步,在2006年9月,我们邀请ACM KDD创新奖得主和IEEE ICDM研究贡献奖得主每人推荐十个最著名的数据挖掘算法,并提供以下信息:

(a) 算法名称;

(b) 算法简介;

(c) 代表文献。

我们还要求每个被提名的算法都应被数据挖掘领域的学者广泛引述和使用,每位推荐人提名的算法集应能代表数据挖掘的不同领域。除一人外其他所有专家都给予了回复。

遴选过程第二步,在2006年10月,我们用Google Scholar对每项提名进行了验证,去除了引用数低于50的提名,将保留下的所有提名(共18个)分成十个主题:关联分析、分类、聚类、统计学习、装袋推举、序列模式、集成挖掘、粗糙集、链接挖掘和图挖掘。对于某些算法,如k-means,不要求提供发明该算法的原始文献,但需要提供阐述该算法重要性的近期论文。可从ICDM站点(http://www.cs.uvm.edu/~icdm/algorithms/CandidateList.shtml)上找到这些代表性文献。

遴选过程第三步,我们动员了研究社区的很多人参与,其中包括KDD-06 (the 2006 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining)、ICDM'06 (the 2006 IEEE International Conference on Data Mining)和SDM'06 (the 2006 SIAM International Conference on Data Mining)的程序委员会的全体委员,以及ACM KDD创新奖得主和IEEE ICDM研究贡献奖得主。请每位参与人员从18个候选算法中选出不超过10个最知名算法,结果在ICDM'06的"数据挖掘十大算法"专题研讨会上公布。

2006年12月21日,在ICDM'06的一个专题讨论会上,邀请145名与会专家对这18个候选算法公开投票,从中选出十个得票最高的算法,得到和上面遴选第三步完全一致的结果。这个3小时的专题研讨会是ICDM'06的一个环节,在同一地点并行召开的还有Web Intelligence(WI'06)和Intelligent Agent Technology(IAT'06)的共7个论文展示环节,共吸引到了145名学者参与。

在ICDM'06之后,我们邀请了这十大算法的原创作者和专题研讨会部分发言人共同撰写了一篇期刊论文对每个算法的内容、影响进行介绍,对其现状和未来趋势加以评述。这篇期刊论文于2008年1月发表在*Knowledge and Information Systems*[1]上。本书是该期刊论文的扩展,每章介绍一个算法,内容包括算法描述、可用软件、示例应用、高级主题和习题等部分。

本书的每一章都邀请两位独立审稿人和本书的一位编辑来审核,有的章节在此基础上

还要在最终定稿前再重审一遍。

我们希望这十个算法的遴选能有助于在世界范围推动数据挖掘的应用，激励更多数据挖掘领域的学者去扩大这些算法的影响，探索新的研究内容。这十个算法覆盖了分类、聚类、统计学习、关联分析和链接分析等重要的数据挖掘研究和发展主题，也对数据挖掘、机器学习和人工智能等学科的课程设计有指导意义。

致　谢

遴选十大数据挖掘算法缘起于2006年5月曹建农博士和吴信东博士在香港理工大学的一次学术研讨。此间，吴博士做了《数据挖掘研究十大挑战》的报告[2]，之后，Vipin Kumar博士又在KDD-06大会上广邀众学者就算法遴选工作进行了探讨，一时应者云集。

明尼苏达大学计算机科学与工程系职员Naila Elliott为三轮遴选过程收集编纂了算法提名和投票结果。佛蒙特大学计算机科学系职员Yan Zhang负责将十份不同格式的提交文档转换成同一LaTex格式，这是一个非常耗时的工作。

参考文献

[1] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. Top 10 algorithms in data mining, *Knowledge and Information Systems*, 14(2008), 1: 1-37.

[2] Qiang Yang and Xindong Wu (Contributors: Pedro Domingos, Charles Elkan, Johannes Gehrke, Jiawei Han, David Heckerman, Daniel Keim, JimingLiu, David Madigan, Gregory Piatetsky-Shapiro, Vijay V. Raghavan, RajeevRastogi, Salvatore J. Stolfo, Alexander Tuzhilin, and Benjamin W. Wah). 10 challenging problems in data mining research, *International Journal of Information Technology & Decision Making*, 5, 4(2006), 597-604.

关于作者

吴信东(Xindong Wu)教授 英国爱丁堡大学人工智能学博士,任美国佛蒙特大学计算机科学系主任。吴教授在数据挖掘、知识系统和 Web 信息开发等研究领域内颇有建树,在 IEEE TKDE、TPAMI、ACMTOIS、DMKD、KAIS、IJCAI、AAAI、ICML、KDD、ICDM 和 WWW 等学术会议和期刊上发表了 170 余篇学术论文,另外,还出版了 18 部学术专著和会议文集。他还获得了 IEEE ICTAI-2005 的最佳论文奖和 IEEE ICDM-2007 的最佳理论/算法论文奖亚军。

吴博士是 *IEEE Transactions on Knowledge and Data Engineering* (TKDE,由 IEEE Computer Society 主办)的主编,*IEEE International Conference on Data Mining* (ICDM)的创始人和指导委员会主席,*Knowledge and Information Systems* (KAIS, 由 Springer 发行)的创办人和荣誉主编, IEEE Computer Society Technical Committee on Intelligent Informatics (TCII)的创始主席(2002—2006), *Springer Advanced Information and Knowledge Processing* (AI&KP)系列著作的编辑。他还是 ICDM'03 (the 2003 IEEE International Conference on Data Mining)程序委员会主席和 KDD-07 (the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining)程序委员会联合主席。他获得了 2004 ACM SIGKDD 服务奖、2006 IEEE ICDM 杰出服务奖,是 2005 年合肥科技大学"长江学者奖励计划"讲座教授。他还是很多学术会议的特邀专家/专题报告人,如 NSF-NGDM'07、PAKDD-07、IEEE EDOC'06、IEEE ICTAI'04、IEEE/WIC/ACM WI'04/IAT'04、SEKE 2002 和 PADD-97 等。

Vipin Kumar 教授 明尼苏达大学计算机科学与工程系 William Norris 讲席教授、系主任。他于 1977 年获得印度鲁尔基理工学院(正式名称是鲁尔基大学)的电子和通信工程学士学位,1979 年获得荷兰埃因霍温飞利浦国际学院的电子工程硕士学位,1982 年获得马里兰大学帕克分校的计算机科学博士学位。Kumar 教授的研究兴趣主要集中在数据挖掘、生物信息学和高性能计算领域。他提出了评估并行算法可扩展性的恒等效率度量指标,并研发了多款稀疏矩阵分解(PSPASES)和图剖分(METIS, ParMetis, hMetis)的高效并行算法及软件。他发表了 200 多篇研究论文,合编合著了 9 本学术专著,包括被广泛使用的教科书 *Introduction to Parallel Computing* 和 *Introduction to Data Mining*,都由 Addison-Wesley 出版。Kumar 是众数据挖掘和多并行计算领域的学术会议、专题研讨会的主席或共同主席,如 *IEEE International Conference on Data Mining* (2002)、*International Parallel and Distributed Processing Symposium* (2001) 和 *SIAM International Conference on Data Mining* (2001)。Kumar 是 SIAM International Conference on Data Mining 指导委员会共同主席,IEEE International Conference on Data Mining 和 IEEE International Conference on Bioinformatics and Biomedicine 指导委员会委员。Kumar 是 *Journal of Statistical Analysis and Data Mining* 的创始主编之一, *IEEE Intelligent*

Informatics Bulletin 主编和 *Data Mining and Knowledge Discovery* 系列图书(由 CRC Press/Chapman Hall 出版)的编辑。Kumar 还担任很多其他学术刊物的编辑,如 *Data Mining and Knowledge Discovery*、*Knowledge and Information Systems*、*IEEE Computational Intelligence Bulletin*、*Annual Review of Intelligent Informatics*、*Parallel Computing*、*Journal of Parallel and Distributed Computing*、*IEEE Transactions of Data and Knowledge Engineering* (1993—1997)、*IEEE Concurrency* (1997—2000)和 *IEEE Parallel and Distributed Technology* (1995—1997)等。他是 ACM 会士、IEEE 会士、AAAS 会士和 SIAM 会员。Kumar 由于在并行算法设计、图剖分和数据挖掘领域的杰出贡献,获得了 2005 IEEE Computer Society 的技术成就奖。

贡 献 人 员

陈松灿，南京航空航天大学，中国
Joydeep Ghosh，得克萨斯大学奥斯汀分校，得克萨斯州奥斯汀
David J. Hand，帝国理工学院，英国伦敦
Alexander Liu，得克萨斯大学奥斯汀分校，得克萨斯州奥斯汀
刘兵，伊利诺依大学芝加哥分校，伊利诺伊州芝加哥
Geoffrey J. McLachlan，昆士兰大学，布里斯班，澳大利亚
Hiroshi Motoda，大阪大学 ISIR 研究所和 AFOSR/AOARD 空军研究实验室，日本
Shu-Kay Ng，格里菲斯大学，澳大利亚梅多布鲁克
Kouzou Ohara，大阪大学 ISIR 研究所，日本
Naren Ramakrishnan，弗吉尼亚理工，弗吉尼亚州布莱克斯堡
Michael Steinbach，明尼苏达大学，明尼苏达州
Dan Steinberg，Salford Systems 公司，加利福尼亚州圣地亚哥
陈封能，密歇根州立大学，密歇根州东兰辛
薛晖，南京航空航天大学，中国
杨强，香港科技大学，香港九龙清水湾
Philip S. Yu，伊利诺依大学芝加哥分校，伊利诺伊州芝加哥
俞扬，南京大学，中国
周志华，南京大学，中国

第 1 章　C4.5

Naren Ramakrishnan

1.1　引言
1.2　算法描述
1.3　算法特性
　　1.3.1　决策树剪枝
　　1.3.2　连续型属性
　　1.3.3　缺失值处理
　　1.3.4　规则集诱导
1.4　软件实现
1.5　示例
　　1.5.1　Golf 数据集
　　1.5.2　Soybean 数据集
1.6　高级主题
　　1.6.1　二级存储
　　1.6.2　斜决策树
　　1.6.3　特征选择
　　1.6.4　集成方法
　　1.6.5　分类规则
　　1.6.6　模型重述
1.7　习题
参考文献

1.1 引 言

C4.5 算法[30]是机器学习和数据挖掘领域中的一整套用于处理分类问题的算法。该算法是有监督学习类型的，即：给定一个数据集，所有实例都由一组属性来描述，每个实例仅属于一个类别，在给定数据集上运行 C4.5 算法可以学习得到一个从属性值到类别的映射，进而可使用该映射去分类新的未知实例。如图 1.1 中所示的例子：Day 列是日期序号，几个属性列(Outlook，Temperature，Humidity，Windy)描述了天气条件，Play Gaof? 列对应于类别(是否适合打高尔夫)。图 1.1 中的表格显示的是“训练数据”——每行是一个实例，实例的属性包括：Outlook(光照，三值)、Temperature(温度，连续值)、Humidity(湿度，连续值)、Windy(是否有风，二值)；实例的类别 PlayGolf? 是布尔类型。C4.5 算法用这个训练数据可以学到一个映射，该映射以新实例(其类别未知)的属性值作为输入，输出是对这个实例所属类别的预测。

Day	Outlook	Temperature	Humidity	Windy	Play Golf?
1	Sunny	85	85	False	No
2	Sunny	80	90	True	No
3	Overcast	83	78	False	Yes
4	Rainy	70	96	False	Yes
5	Rainy	68	80	False	Yes
6	Rainy	65	70	True	No
7	Overcast	64	65	True	Yes
8	Sunny	72	95	False	No
9	Sunny	69	70	False	Yes
10	Rainy	75	80	False	Yes
11	Sunny	75	70	True	Yes
12	Overcast	72	90	True	Yes
13	Overcast	81	75	False	Yes
14	Rainy	71	80	True	No

图 1.1 C4.5 算法所用数据集的一个示例

J. Ross Quinlan 设计的 C4.5 算法源于名为 ID3 的一种决策树诱导算法[25]，而 ID3 是被称为“迭代分解器(iterative dichotomizers)”系列算法的第 3 代。决策树相当于将一系列问题组织成树，具体说，每个问题对应一个属性(比如 Outlook)，根据属性值来生成判断分支，一直到决策树的叶节点就产生了类别(此处，就是 PlayGolf?)的预测结果。决策树和你利用的车辆用户手册排查你的车到底出了什么问题没什么不同。C4.5 算法除了能诱导出决策树，还可以将决策树转换成某种具有良好可理解性的规则。特别是进一步看到，通过 C4.5 的后剪枝操作得到的分类器不能再精确地被转换回决策树。

C4.5 算法的发展历史非常生动地展现了不同的学术社区在分类问题研究上的殊途同

归的情形。一方面，ID3 算法和 Original Tree 算法是各自独立发展的，而 Original Tree 算法最初由 Friedman 发明[13]，后来在 Breiman、Olshen 和 Stone 等人的参与下发展成 CART 算法[4]。另一方面，可以看到（从 ID3 发展而来的）C4.5 算法也大量引用了的 CART 算法的文献[30]，其设计理念很大程度上也受到 CART 算法的影响，例如对特殊类型属性的处理方式（为了尽量避免书中不同章节之间内容的重叠，本书在第 10 章中专门对 CART 算法进行介绍，而本章则尽可能缩减关于 CART 算法的论述，仅在必要处点出二者的关联）。此外，在文献[25]和[36]中，Quinlan 还肯定了 CLS（概念学习系统[16]）框架对于发展 ID3 算法和 C4.5 算法的重要作用。如今，经典的 C4.5 已经被 See5/C5.0 所取代，后者是 Rulequest Research 公司的商用产品。

数据挖掘社区遴选出的十大算法中有两个都是基于树的算法，这也反映出此类算法在数据挖掘中被使用的广泛程度。最早的决策树仅处理标称/类别的数据类型，而如今已扩展到支持数值、符号乃至混合型的数据类型。具体的应用领域也很广泛，例如临床决策、生产制造、文档分析、生物信息学、空间数据建模（地理信息系统）等。实际上，只要是目标问题的类间边界能用树型分解方式或规则判别方式来确定，就可以使用 C4.5 算法。

1.2　算法描述

我们谈到 C4.5 时通常并不是指一个单一的算法，而是泛指基本的 C4.5、C4.5-no-pruning 以及拥有多重特性的 C4.5-rules 等诸多变体的一整套算法。下面先介绍最基本的 C4.5 算法，然后再进一步讨论其特性。

算法 1.1 在从宏观上描述了 C4.5 的工作原理。所有的树诱导方法都大体上遵循一种统一的递归模式，即：首先，用根节点表示一个给定的数据集；然后，从根节点开始在每个节点上测试一个特定的属性，把节点数据集划分成更小的子集，并用子树表示；该过程就要一直进行，直到子集成为“纯”的，就是说直到子集中的所有实例都属于同一个类别，树才停止增长。

算法 1.1　C4.5(D)

```
Input: an attribute-valued dataset D
1: Tree={}
2: if D is "pure" OR other stopping criteria met then
3:    terminate
4: end if
5: for all attribute a∈D do
6:    Compute information-theoretic criteria if we split on a
7: end for
8: a_best = Best attribute according to above computed criteria
9: Tree = Create a decision node that tests a_best in the root
10: D_v = Induced sub-datasets from D based on a_best
11: for all D_v do
12:    Tree_v = C4.5(D_v)
13:    Attach Tree_v to the corresponding branch of Tree
14: end for
15: return Tree
```

图 1.1 给出的是用于演示 C4.5 算法的 golf 数据集，该数据集已被打包在 C4.5 软件的安装程序中了。如前所述，该示例的目的是演示根据给某日的天气状况来预测该日是否适合打高尔夫球。要注意在这个示例中，有些特征值是连续型，而另一些是类别型。

图 1.2 显示的是基于图 1.1 中的“训练数据”用 C4.5 算法（采用默认选项）诱导出的树。下面介绍从数据中诱导出这样的树所面临的各种选择。

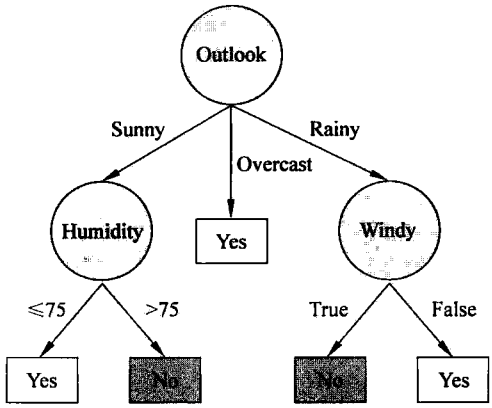


图 1.2 C4.5 算法基于图 1.1 的数据集获得的决策树

- 有哪些可能测试类型？

如图 1.2 所示，C4.5 算法并不限定只能进行二元测试，也允许有多于两项输出的测试。如果属性为布尔型，测试就会诱导出两个分支；如果属性是类别型，就需要进行多值测试；当然，也可以将这些值分组以生成更少的选项，此时每个选项相当于一个属性值，如此即可用这些选项来进行预测。如果属性是数值型的，测试就是{≤θ? 或>θ?}形式的二元值，其中 θ 是属性的一个合适的判定阈值。

- 如何对测试进行选择？

C4.5 算法使用增益（gain）、增益率（gain ratio）等信息论准则来对测试进行选择。增益被定义为“执行一个测试所导致的类别分布的熵的减少量”，增益准则的一个缺陷在于她过于偏向选择具有更多输出结果的测试，而增益率具有克服这一偏差的优点，所以 C4.5 算法默认的测试选择准则是增益率。在树增长的每一步中，C4.5 算法都要选择具有最符合准则的那个测试。

- 如何选取测试的阈值？

如前所述，对于布尔型和类别型的属性，测试所用的值也就仅仅是该属性的可能取值。但对于数值型属性，相应测试就需要确定阈值，就是要对该属性的取值进行排序进而求出合适的切分点以最大化上述测试选择准则。Fayyad 和 Irani[10]的研究显示不是所有的前后相继的值都需要被考虑在内，例如：对于一个连续型属性的两个相继的值 V_i 和 V_{i+1}，如果这两个取值对应的实例属于同一类别，那么将它们分裂开来是不能提高信息增益（或增益率）的。

- 如何决定停止树生长？

通常，如果某节点的一个分支所辖的全部实例都是“纯”的，则这个分支就被确定为一个叶节点；另一种可能的终止条件是，如果该分支覆盖的实例总数已经低于预定的阈值。

• 如何确定叶节点类别?

叶节点包含的实例中,占据最大比例的那个类别将被作为该分支的类别。

这些问题是所有依托于树的分类方法和绝大多数基于树诱导算法的决策方法都需面对的共性问题。在基本的树诱导算法之上,C4.5 又引入了一组额外的特性,正是这些新的特性确立了 C4.5 算法的实际效用。先用如图 1.1 所示的简单数据集为例来阐述一下基本的树算法 1.1,才好进一步讨论 C4.5 算法引入的新特性。

下面仔细研究一下图 1.2 所示的树是如何从图 1.1 所示的数据集转化来的。可以观察到 Outlook 被选为第一个进行测试的属性,那为什么会这样呢? 需要先计算类别随机变量(PlayGolf?)的熵。该变量是一个二值变量,其取值的概率分别为 9/14(是)和 5/14(否)。我们知道,如果某随机变量有 c 个取值的,其对应概率为 $P_1, P_2, \cdots, P_c$,则该随机变量的熵为:

$$\sum_{i=1}^{c} -p_i \log_2 p_i$$

所以,PlayGolf? 的熵值可以这样计算:

$$-(9/14)\log_2(9/14)-(5/14)\log_2(5/14)$$

得结果为 0.940。用信息论的原理讲,为了能确定随机变量 PlayGolf? 的取值,平均需要获得关于该随机变量的 0.940 位的信息。在树诱导过程中,C4.5 算法的目标就通过合适的提问来获得信息,实现这个熵值的下降。我们依次考查每个属性,计算该属性导致的熵值下降幅度。我们称这种下降幅度为信息增益(Gain),如此处的属性 Outlook 对应的信息增益可记为 Gain(Outlook),具体可以通过下面公式进行计算:

$$\text{Entropy(PlayGolf? in D)} - \sum_{v} \frac{|D_v|}{D}\text{Entropy(PlayGolf? in } D_v)$$

式中的 v 是目标属性的所有可能取值(具体到此处就是 Outlook 的 3 个取值);D 表示整个数据集;D_v 表示 D 的一个子集,其实例在目标属性上同取某个值(具体到此处就是属性 Outlook 的某个取值),符号 $|\cdot|$ 表示目标数据集的大小(就是实例的数量)。

上述计算得到属性 Outlook 可以获得的信息增益 Gain(Outlook)为 0.940−0.694=0.246。利用相同的方法,可以计算出属性 Windy 的信息增益 Gain(Windy)为 0.940−0.892=0.048。类似地,可计算出所有其他属性的信息增益值,经比较后会得出属性 Outlook 的信息增益最大,因而属性 Outlook 被选为用于产生树的第一个分支的最佳属性。在此,我们认真观察一下就发现这是一种贪婪选择,而且没有考虑到未来决策对该选择的影响。前面讲到树生长的停止准则,其中一种就是一个树会持续生长,直到分裂所得的子数据集变成"纯"的。在此处的例子中,属性 Outlook 的取值 Overcast 所对应的那个分支的数据集就是"纯"的,即该分支上所有实例的类别变量 PlayGolf? 的值都是 Yes。因此,在这个分支下树就不再生长了。但是,注意到属性 Outlook 的另外两个取值生成的都是"非纯"数据集。接下来算法递归,继续选择属性,但是不能再选属性 Outlook 了(为什么?)。通常,选择不同的测试属性就会产生不同的数据集分裂,最终导致树生成不同的分支,但是有的数据集仍有可能产生重复子树(此处原作者表达不清晰,应该是说,为避免树的父子节点出现重复的情况,最直接的办法就是不能选已经用过的属性——译注)。

前面提到 C4.5 算法的默认分裂准则实际是信息增益率(Gain Ratio),而非信息增益

(Gain)。为了搞明白二者的不同,假定图 1.1 的 Day 列是一个可用于分类的"真实"特征。更进一步,可将其当成一个具有标称值的属性。当然了,由于每天是唯一的,所以 Day 实际上对分类没有任何作用,所以没必要将其算作属性。不过,如果我们硬要将她当成可用属性的话就会产生一个不合理的结果:该数据集中 Day 有 14 个不同的值,每个值都诱导出一个"纯"的数据集(仅含有 1 个实例的平凡数据集),这样 Day 就会被选为树生成分支的最佳属性!那为什么会产生这样一个荒谬的结果呢?这是因为信息增益 Gain 的定义决定了她偏向于取值多的属性。于是,Quinlan 提出了用信息增益率来校正这一负面效应。一个属性 a 的信息增益率被定义为:

$$\text{GainRatio(a)}=\frac{\text{Gain(a)}}{\text{Entropy(a)}}$$

观察这个公式就会发现:属性 a 的熵 Entropy(a)仅决定于她的取值的概率分布,与类别无关;而属性 a 的信息增益 Gain(a)是与类别相关的(此外,虽然公式中没有明确标示,但我们要知道这里所有的计算都依赖于所用的数据(子)集)。用以上公式可以算出属性 Outlook 的信息增益率 GainRaito(Outlook)=0.246/1.577=0.156。用相同的方法,可以算出其他所有属性的信息增益率。那么要问属性 Outlook 是否应被选为在根节点处做决策的测试呢?读者可将此问题作为一个练习试试看。

讨论到目前这个阶段,我们应该强调一下决策树建模并不是对所有类型的决策边界都适用。以布尔函数为例,虽然原则上决策树可以建模任意布尔函数,但是最终得到的树会变得非常复杂。进一步再考虑一个更具体的情形,就是对大量布尔属性的 XOR 操作建模。这种情况下,每个属性都会出现在树的所有路径上,最终导致树的规模成指数级增大。决策树难以处理的另一个问题就是所谓的"m-of-n"函数。该问题是说要用 n 个属性中的 m 个进行类别预测,但是我们并不确切知道是哪 m 个属性对决策有贡献。针对这个问题,已经有一些如斜决策树这样的解决方案(将在本文的后面部分介绍)。除了这些难点,C4.5 算法在创建决策树过程中还引入了子树复制问题,根源在于该算法遵循的属性选择的贪婪策略。通常情况下,必须通过穷尽搜索才能获得最佳属性,这就要求生长出完全的树。

1.3 算法特性

1.3.1 决策树剪枝

为了避免生成的树过度拟合训练数据,必须对树进行剪枝处理。为了充分说明这个问题,Quinlan 给出了一个极端的数据集[30],它包含 10 个布尔属性,取 0 或 1 是等可能的;类别变量为二值,取 yes 的概率为 0.25,取 no 的概率为 0.75;数据集共含 1000 个实例,取其中 500 个训练,剩下的 500 用于测试。Quinlan 发现 C4.5 算法生成一个多达 119 个节点的树!但是,该树的错误率居然高过一个更简单的树 35%还多。由此可见,剪枝对于提高树对新实例类别预测的准确性至关重要。剪枝过程通常是在树全部生成后进行,而且采用自下而上的方式。

Quinlan 在 1986 年麻省理工人工智能实验室(MIT AI lab)的备忘录中概述了此前学

者们提出的多种可行的树剪枝方法。在另一种著名的决策树归纳算法 CART 中，采用了一种被称为“基于成本复杂度的剪枝(cost-complexity pruning)”的方法，这种方法会生成一系列树，每棵树都是通过将前面的树的某个或某些子树替换成一个叶节点而得到的，系列中的最后一棵树仅含一个用来预测类别的叶节点。成本复杂度是一种度量准则，用来判断哪棵子树应该被一个预测类别值的叶节点所代替。这种方法要使用一个单独的测试数据集来评估所有的树，根据它们在测试数据集上的分类性能选出“最佳”的树。

错误消减剪枝(Reduced error pruning)是以上方法的一个简化，和先前的方法一样的地方是该方法也是使用一个单独的测试数据集，不同之处在于她直接使用完全诱导树对测试集中的实例进行分类。对于诱导树中的每一个非叶子树，该策略要求评估用一个最佳叶节点去替代这棵子树的是否有益。如果树在剪枝后与在剪枝前相比，其错误率是保持或下降的，而且被剪掉的子树也不包含具有相同性质的其他子树，则这棵子树就可以被替换掉。这个过程一直持续到在测试数据集上树的分类错误率出现上升为止。

C4.5 算法创造性地提出了被称为悲观剪枝(Pessimistic pruning)的方法，该方法不再需要一个单独的测试数据集，而是通过在训练数据集上的错误分类数量来估算未知实例上的错误率。悲观剪枝方法通过递归计算目标节点的分支的错误率来获得该目标节点的错误率。例如，对有 N 个实例和 E 个错误(就是说所属类别与该叶节点预测类别不一致的实例数量)的叶节点，悲观剪枝首先用比值(E+0.5)/N 来确定叶节点的经验错误率。设一个子树有 L 个叶节点，并且这些叶节点共包含 $\sum$E 个错误和 $\sum$N 个实例，则该子树的错误率可估算为 $(\sum E + 0.5 * L)/\sum N$。假设该子树被它的最佳的叶节点替代后，在训练数集上得到的错误分类的数量为 J。那么，如果(J+0.5)在 $(\sum E + 0.5 * L)$ 的 1 标准差范围内，悲观剪枝方法就决定用该子树的这个最佳叶节点替换这棵子树。

这种方法被扩展成基于理想置信区间(confidence intervals，CI)的剪枝的方法。该方法将叶节点的错误率 e 建模为服从 Bernoulli 分布的随机变量，对于一个置信区间阈值 CI，存在 e 的一个上界 e_{max}，使得 $e < e_{max}$ 以 1−CI 的概率成立(C4.5 对于 CI 的默认值为 0.25)。更进一步，我们可以用正态分布来逼近 e(只要 N 足够大即可)。基于这些约定条件，C4.5 算法的期望误差的上界为：

$$\frac{e + \frac{z^2}{2N} + z\sqrt{\frac{e}{N} - \frac{e^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}} \tag{1.1}$$

式中 z 的选择是基于理想置信区间，假设 z 是一个拥有零均值和单位方差的正态随机变量，也就是 N(0,1)。

最后，还需要说明的是剪枝操作的确切过程，该过程是一个单一的自底向上的遍历。图 1.3 显示的是一个剪枝过程的中间步骤，左半部分所示 X 的子树 T_1、T_2 和 T_3 已经完成剪枝操作了，右半部分所示的三种情况都需要分别估算对应的错误率。第一种情况是保持树的原状；第二种情况是仅保留 X 输出最大的那颗(在这个例子中，是中间分支)；第三种情况是将子树替换为叶节点，其类别选择训练集中的最大类别。以上策略被自底向上执行直到抵达树的根节点。

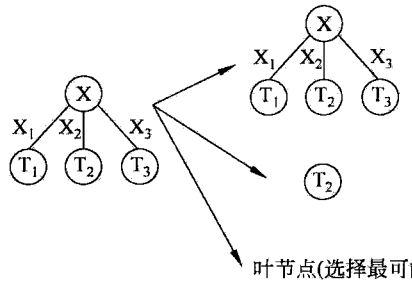


图 1.3 决策树剪枝的不同选择，由上到下左边的树可以保持本来的样子，也可以被它的一棵子树代替，也可以被一个叶子节点代替

1.3.2 连续型属性

Quinlan 的论文[31]中论及很多关于连续变量的复杂处理方法。连续变量相对于离散变量所具有一些优点，例如连续变量能够使用更多种决策准则来产生分支，这自然会给连续变量带来更多优势。一种处理连续变量的方法就是像前面那样使用信息增益率代替信息增益。但是，这样做我们会碰到一个问题，因为信息增益率会受到连续属性所采用的阈值的影响。具体来说，如果选择的阈值导致实例被基本平均分配，那么信息增益率就会趋于极小（因为变量的信息熵处在分母的位置）。因此，Quinlan 建议使用常规的信息增益选取阈值，不过仍要首先使用信息增益率选择属性。第二种处理连续变量的方法基于 Risannen 的最小描述长度（MDL）原理。这种方法把树看成理论，Quinlan 提出应该寻求树的复杂度和分类性能之间的平衡，特别是计算树的复杂度时要综合考虑树的编码成本和树的例外情况（就是树错分的训练实例）。经验表明这种方法不仅限于连续值处理。

1.3.3 缺失值处理

无论是在前面的学习阶段还是在后续的分类新实例的阶段都需要对缺失属性进行特别的处理。Quinlan 文献[28]对与此相关的必须被考虑的多个议题做了非常完备的评述。其中有三个问题比较重要：

（1）生成树分支时要比较多个属性，有些属性在有些实例上没有值，那我们如何选出用于分裂树的合适属性呢？

（2）在为测试选定属性之后，在该属性上没有值的那些训练实例不能放入该测试的任何输出中。但是为了让树的生长能持续进行，又必须让这些实例参与进去。所以，第二个问题就是，当数据集分裂成子数据集时，应该怎么处理这部分属性值缺失的实例？

（3）最后，当用树分类新实例时可能会在树中遇到某个测试，如果该实例没有对应属性值，如何继续推进测试过程？前两个问题涉及诱导树的学习任务，而第三个问题发生在用学习得到的树对新实例进行分类的阶段。人们已经对这几个问题都找到一些可行解决方法。在文献[28]中，Quinlan 为每一个问题都提出了多重解决手段，所以我们可以通过将各个问题的具体解决手段组合起来形成缺失值处理的集成解决方案，Quinlan 为此还设计了一种

编码方案。

第一个问题，是要在含缺失值的属性 a 上进行属性选择准则的计算，我们可以有不同的做法：(I)忽略训练数据中那些在属性 a 上没有值的实例；(C)选用最常用的值(二值型和类别型属性)或者均值(数值型的属性)；(R)对具有缺失值的属性 a 的信息增益/信息增益率依据涉及缺失值的实例的比重来折算或者(S)对训练数据中的缺失值进行“填充”，具体的做法可以是把这种缺失值赋一种独特的新值，或者通过其他已知的属性去尝试确定缺失的值。CART 算法(见第 10 章)所用的代理分裂的思想就是最后一种处理缺失值想法的一个实现。

第二个问题，是关于在递归建立决策树过程中分裂训练集，如果树基于属性 a 建立分支，而属性 a 有一个或多个训练实例存在缺丢失值，我们可以采取以下方法处理：(I)直接忽略这个实例；(C)该实例为这个属性取一个最常用的值；(F)将该实例切分后赋给每个子数据集，不同数据集得到该实例的份额同其他已知属性值的实例的数量成正比；(A)直接将该实例赋给所有子数据集；(U)为属性 a 含缺失值的情况专门建立一条单独的分支或者(S)确定属性 a 的最可能的值(就如前面那样，用文献[28]所说的方法)，然后将其赋给相关的子数据集。在文献[28]中，Quinlan 提出了一种(F)方法的变体，将实例仅赋给一个数据集，赋值的份额同该子数据集里已知属性值的实例数量成比例。

最后，如果在分类的阶段遇到实例的属性 a 存在缺失值的情况，可以采取如下的办法：(U)如果对于属性 a 已经有一个缺失值处理的单独分支，那么就走这个分支；(C)从属性 a 的最常用分支走；(S)运用之前提到的文献[28]的办法来确定属性 a 的最有可能取值，并走相应的分支；(F)同时探查所有的分支，并组合它们的结果来得到不同输出对应的概率[27]；或者(H)不再进行测试，将最有可能的类别赋给该实例。

读者或许已经猜到，有些组合相对更加自然，而也有一些组合没有什么意义。对于按比例分配的处理，只要满足权值之和是 1 的约束，就可以认为是一种来计算信息增益和信息增益率的自然推广。

1.3.4 规则集诱导

C4.5 算法的一个非常突出的特性是其剪枝能力，它建立在诱导树所生成的规则上。我们可以将一棵树看成是多项规则的合成，而每条规则对应着树中从根节点到叶节点的一条路径。规则的前件是该路径上的决策条件，而后件是被预测的类别标签。对于数据集中的每一个类别，C4.5 算法首先依据(未剪枝的)树建立规则集，然后对每一条规则执行一个爬山搜索看是否有规则前件可以被去掉。由于规则前件的移除相当于“敲掉”诱导决策树的一些节点，所以 C4.5 的悲观剪枝方法其实就是用在这里。剪枝会为每一个类别选择一个简化的规则子集。这里用最短描述长度原则(MDL)来刻画编码和排序规则的理论所需成本。生成的规则的数量一般比原始树中的叶节点(或路径)的数量少很多。另外还可以观察到一个现象就是，由于所有的规则前件都有被可移除的可能，这样当靠近树的顶端的节点被移除后，所产生的规则不一定能被还原为一棵紧凑的树。C4.5 规则集生成方法的一个缺点是数据集规模的增大会导致学习时间的迅速增长。

1.4 软件实现

J. Ross Quinlan 本人的 C4.5 算法实现可以在他的个人站点上找到：http://www.rulequest.com/Personal/。要注意的是，该软件是有版权的，因而只有经过作者的授权才可以将其用于商业用途。不过，C4.5 的代码也授权给个人使用，这也促使 C4.5 成为领域的标准。有很多可用的公共领域的 C4.5 实现，例如 Ronny Kohavi 的 MLC++ 库[17]，现在是 SGI 的 Mineset 数据挖掘套件的一部分；还有来自新西兰 Waikato 大学的 Weka[35]数据挖掘套件(http://www.cs.waikato.ac.nz/ml/weka/)。Weka 中对 C4.5 的 Java 实现是被编号为 J48。C4.5 的商业化实现主要包括：来自 Intelligent Systems Research，LLC 公司的 ODBCMINE，该软件基于 ODBC 数据库接口；还有 Rulequest 公司的 See5/C5.0，它在很多方面上改进了原始的 C4.5 算法，当然也包括对 ODBC 的支持。

1.5 示　例

1.5.1 Golf 数据集

现在，我们要在 Golf 数据集上来详细地描述 C4.5 算法的功能，以默认选项运行 C4.5，即：

```
>c4.5 -f golf
```

C4.5 产生下面的输出：

```
C4.5 [release 8] decision tree generator  wed Apr 16 09:33:21 2008
-------------------------------
    options:
        File stem <golf>
Read 14 cases (4 attributes) from golf.data
Decision Tree:
outlook=overcast:play(4.0)
outlook=sunny:
|  humidity<=75: Play(2.0)
|  humidity>75:Don't play (3.0)
outlok=rain:
|  windy=true:Don't play (2.0)
|  windy=false: play (3.0)
Tree saved
Evaluation on training data (14 items):
        Before pruning              After Pruning
        ------------    -------------------------
        Size    Errors      Size    Errors    Estimate
          8     0(0.0%)       8     0(0.0%)    (38.5%)  <<
```

仔细考察 C4.5 的输出，特别是结尾处的统计信息包括未剪枝树和已剪枝树的大小(也

就是节点的数量，包括内部节点和叶节点）；未剪枝树和已剪枝树在训练数据集上的错误率；剪枝后书的错误率估计。在这个例子中，从观察到的情况来看，没有进行剪枝。

C4.5 的-v 选项可以提供详细信息而且能划分多个级别，比如可以给出关于信息增益计算的每一步信息。c4.5rules 这个软件和 C4.5 类似，不同之处在于她能根据可能的后剪枝操作来产生规则，这种方法在前面已经阐述过。在 Golf 数据集上使用默认的选项时不会发生剪枝，输出 4 条规则（对应图 1.2 所示路径，有一条除外）和 1 条默认规则。

诱导树和规则要被用在未曾见过的“测试”数据集上，并以此来评估它的泛化性能。C4.5 利用-u 选项指定测试数据，进而评估诱导树/规则的性能。

1.5.2 Soybean 数据集

Michalski 的 Soybean 数据集是一个经典的机器学习测试数据集，它来自 UCI 机器学习资料库[3]。该数据集涉及 35 个属性，总共包括 307 个实例，而且有很多的缺失值。下面是 UCI 网站的相关描述：

> 该数据集总共有 19 个类，但早期的研究工作只用到前 15 个类。有一种非正式的说法认为，排在后面的 4 个类都还没有得到确认，因为这几个类的实例太少了。该数据集涉及的属性共 35 个，有些是标称型的，有些是有序的，dna 用来表示属性不可用。属性的值都是数字化编码的，第一个值编码为 0，第二个为 1，以此类推，未知的值编码为“?.”。

在这个数据集上学习决策树的目的是为了基于观察到的大豆形态特征来辅助大豆疾病的诊断。

该数据上生成的诱导树非常复杂，所以，我们在此处仅描述树在剪枝前后的大小和性能：

```
Before Pruning|          After Pruning
--------------      ---------------------
Size    Errors      Size    Errors    Estimate
 177    15(2.2%)     105    26(3.8%)    (15.5%)  <<
```

从这里可以看到，未剪枝的树在训练数据集上的分类性能不是非常好，并且在完成树的诱导之后进行了明显的剪枝。为了确认分类器是“最终”的版本，必须使用诸如交叉验证之类的严格评估过程。

1.6 高级主题

在基于树/规则的分类器这一特定领域中有很多有趣的前沿问题，而整个现代数据挖掘的研究中，一个非常重要的方面是对海量数据的处理。在这里我们讨论一部分相关的内容，而另外一部分留到后面的习题中。在 KDD、ICDM、ICML 和 SDM 等学术会议的文集中，可以看到这些领域最新进展。

1.6.1 二级存储

KDD 社区研究中所用的很多数据集都非常大，不能被完全装入内存中，因此，为了能够对二级存储设备上的数据进行有效处理，就不得不对机器学习算法进行重新设计和实现。这方面一个关键的算法设计问题是，在确保诱导出的分类器效能的前提下，如何做到数据集扫描遍数的极小化？BOAT 算法[14]基于自举的策略，首先是抽样，即将原始数据集分成的很多小的子集以便能放入内存中，从而可以生成很多树；然后是融合，即将这些树相互叠加得到符合“粗粒度”的分裂准则的一棵树；最后，再通过完整扫描一遍原始数据集，就可以精化并得到最终的分类器。RainForest 框架[15]采取集成的策略，她实现了多种具体的决策树构建方法，该框架具有良好的可扩展性，适用于大规模数据集的处理。其他基于二级存储设备的挖掘算法还有 SLIQ[21]、SPRINT[34]以及 PUBLIC[33]等。

1.6.2 斜决策树

斜决策树适用于处理连续值数据，在对应坐标系下，其决策边界可以处于任意位置且可以具有任意倾角（参考后面的练习 2），故而得名斜决策树。举个例子，如果要替换属性 a_1 上的决策准则 $a_1 \leqslant 6$?，我们可能会在决策树的一个节点上使用涉及两个属性的准则，比如 $a_1 - 2a_2 \leqslant 6$?采用属性线性组合策略的一个典型的决策树分类器就是 Murthy、Kasif 和 Salzberg 等人研发的 OC1 系统[22]，CART 算法是 OC1 的重要基础。其基本思路是先进行轴平行分裂，再通过“扰动”改进分裂效果。首先，将轴平行分裂转换为属性值的一个线性组合，然后迭代地调整该线性组合的系数从而获得更好的决策准则。显而易见，在这种情况下，诸如错误估计、剪枝和缺失值处理等方方面面都必须重新考虑。实际上，OC1 是爬山法和随机化系数微调法的精巧组合。还有其他一些诱导斜决策树的方法，例如文献[5]。

1.6.3 特征选择

到目前为止，我们还没有特别强调特征选择对于基于树/规则的有监督学习任务的重要性。有一些特征可能和类别预测没有关系，也有一些特征可能是多余的。特征选择说的就是通过对特征集合进行约减而得到一个更小的特征子集。有一些特征选择方法是需要和特定学习算法协同使用的，也有一些和学习算法无关的特征选择算法，如 Koller 和 Sahami[18]描述的方法。

1.6.4 集成方法

集成方法已经成为了机器学习和数据挖掘的主流方法之一。而其中装袋法与推举法（见第 7 章）是两种是最流行的集成方法。装袋法对训练数据进行随机重采样，为每个采样诱导出一棵树。然后再使用表决等方法将所有树的预测结果组合成一个输出。在推举法中[12]（见第 7 章）会产生一系列的分类器，其中每一个的训练数据都是依赖于前一个步骤所得到的分类器。特别需要说明的是，不能被某一步分类器正确预测那些实例在下一个步骤中将会被赋予更多的权重。最终的预测结果是从所有单个的分类器的预测结果汇总得出的。C5.0 系统支持一种推进法变体，她构建了集成分类器，可以通过表决得到最终的分类

结果。Opitz 和 Maclin[23]讨论了多种决策树集成方法并和神经网络进行了比较。而 Dietterich[8]则比较了集成方法和"随机化"方法，后者是指学习算法生成的内部决策时引入了随机性。可替换决策树算法(alternating decision tree algorithm)[11]将树生长和推进紧密结合在一起，除了用作测试条件的节点之外，可替换决策树还引入了所谓的"预测节点"，这种节点附带一个分值，具体的计算要沿着从根到叶节点决策路径的方向。实验的结果表明它和推进决策树一样稳健。

1.6.5 分类规则

在获得类似于 C4.5 算法的分类规则的研究中，通常存在两种截然不同的路线。我们可以按照其起源将其大致分为预测性分类器和描述性分类器。但是，最近的研究已经使它们的界限变得模糊了。

遵循预测性路线的算法包括了诸如 CN2[6]和 RIPPER [7]等。这些算法不论是自底向上还是自顶向下，通常都要是采取"顺序发现"形式，即挖掘出一条新规则后，就要将该规则覆盖的实例从训练集中移除，然后再挖掘新的规则，以此类推。自底向上的方法是从实例(叶节点)出发，通过关联单个实例的属性值和类别值来获取规则。规则是多个属性的合取，需要将某些属性有组织地移除，从而考查规则的预测精度。通常不用全局搜索，而是用到局部搜索，如柱搜索(beam search)算法。当该规则被添加后，她所覆盖的例子都要被移除，新的规则从剩余的训练数据中导出。类似地，自顶向下的方法从前件为空的规则(根节点)开始，综合待预测类别值和属性测试来确定一条合适的规则。

描述性路线源自于关联规则的研究，关联规则是 KDD 社区一种流行的技术[1,2](见第 4 章)。习惯上来说，关联存在于两个项集(记为 X 和 Y)之间，一般用 X→Y 来表示。关联的度量指标主要有支持度(全数据集中同时含有 X 和 Y 的实例的百分比)和置信度(含有 X 的实例中，含有 Y 的实例所占的百分比)。关联规则挖掘的目的就是要找到满足给定支持度和置信度的阈值的所有关联。CBA(基于关联规则的分类)[20]是关联规则挖掘在分类问题中的应用，其目标就是确定与特定类别一致的全部关联规则，然后再用这些规则去构建分类器。CBA 的剪枝和 C4.5 的错误估计方法类似，而二者的主要不同之处是 CBA 要对所有可能的规则的进行穷举搜索，而且 CBA 将传统的关联规则挖掘算法改进为高效的(决策树)规则挖掘。当前，描述性路线下的研究工作在 KDD 社区中非常活跃，产生了一批新的方法变体和实际应用。

1.6.6 模型重述

文献[32]中首次引入了重述这一概念，重述的实质是对规则的等价推广。重述从字面意思讲就是重新描述，展开说就是用不同的词汇来表达同一概念。给定一个用于描述的词汇表，重述挖掘的目的就是要用里面的词汇来构建两个不同的表达式，而他们诱导出的是同一个对象子集。这里有个基本的假设：这种集合(至少)可以用两种方法确定，并且其最终行为是一致的，这也正是重述有趣的地方。

重述挖掘算法 CARTwheels 可以生成两棵"类 C4.5 树"，这两棵树生成方向相反且在叶节点层实现匹配(一致)。实际上，其中的一棵树通过选择一些特定的子集来实现对象的

划分，而另一棵树则选用不同的子集，但两棵树生成的划分是匹配的。如果划分一致性成立，那么只需将树的路径合并就可以得到重述。CARTwheels 算法实质是对树匹配的空间进行搜索，即通过一个交替过程，不断地生长新树并匹配由另一棵树生成的划分。重述挖掘已经在很多方向上得到了推广[19,24,37]。

1.7 习 题

1. 请对 C4.5 算法给出的决策树诱导过程的大 O 时间复杂度进行细致的量化，给出关于属性数量和训练实例数量的复杂度。可以考虑先确定树深度的界，再据此得出构建树所需的时间。此外，也请给出对剪枝代价的评估。

2. 请设计一个数据集，令其包含连续值属性并且相应决策面不平行于坐标轴。在该数据集上运行 C4.5 算法，并对诱导出的决策树进行评估，实验中需考虑精度(accuracy)、树的大小和可理解性等因素。

3. 避免过拟合的另一种办法是限制树的生长(而不是在树完全长成后再将其修剪到一个小的尺寸)。考虑一下这种预剪枝的想法是否合理？以及为什么？

4. 请证明 C4.5 算法所用的不纯度(也就是熵)度量是凹的。为什么不纯度的凹性是重要的？

5. 请推导等式(1.1)，如文中所写，对建模错误率的贝努力随机变量用正态分布进行近似。

6. 请思考如果直接用预测精度(而不是信息增益)来选择属性，那么会对决策树的诱导产生什么样的影响？进而，如果归纳仅含有一个前件的规则又如何？提示：请参考 Robert Holte 的工作“R. Holte, Very Simple Classification Rules Perform Well on Most Commonly Used Datasets, Machine Learning, vol. 11, pp. 63-91, 1993.”

7. 在某些机器学习应用中有的属性具有“集合值”，例如一个对象可能同时包含多种颜色，将其颜色属性建模为“集合值”(而非“实例值”)对该对象的预测任务可能是非常有益的。请寻找一些例子来说明基于“集合值”属性的决策测试，并说明 C4.5 算法的决策树生长过程是很容易集成“集合值”属性的。

8. 有的场景不是要将一个实例分到某个类中，而是要根据该实例在各个类别上的归属(后验)概率对类别进行排序。参考文献“Read F. Provost and P. Domingos, Tree Induction for Probability Based Ranking, Machine Learning, vol. 52, no. 3, pp. 199-215, 2003”解释了为什么 C4.5 算法归纳出的决策树不适合用作可信概率估计，该文献也提出了一些概率平滑方法来修正这一问题。请思考一下 C4.5 规则是否也有这种问题，是否也能使用这种解决方案？可以使用 UCI 机器学习资料库做一些实验。

9. (取自：S. Nijssen and E. Fromont, Mining Optimal Decision Trees from Itemset Lattices, Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 530-539, 2007.)C4.5 决策树归纳算法的目标是要识别出最优的树，而最有性可以有多种考量，例如：(1)不超过某最大深度的预测最精确的树；(2)每个叶节点覆盖至少k 个实例且在未知样本上具有最大期望预测精度的最小的树。请描述归纳出这两种最优树的高效算法。

10. 一阶逻辑比本章讨论的属性-值表示方法具有更丰富的表达能力。给定一个一阶关系的集合，请描述如何改进 C4.5 算法，使其可以使用这些一阶特征。请注意解决方案必须能诱导如下形式的树或规则：

grandparent(X,Z) :- parent(X,Y), parent(Y,Z)

这里是说，X 是 Z 的祖父的条件是：存在 Y 是 X 的父亲且 Z 是 Y 的父亲。用一阶逻辑当做表示语言会产生几个新问题，如：第一，和属性-值的情况不同，一阶逻辑特征（如 parent(X,Y)所示）并不是事先给定的而是需要从特定实例中泛化；第二，如果规则头的变量在规则体里没有出现就可能会生成无意义的树或规则，例如：

grandparent(X,Y) :- parent(X,Z)

请描述如何将检查和平衡导入诱导过程，使得可以将完备的一阶理论用于数据诱导。提示：可以参考归纳逻辑编程领域的思想[9]，特别是一些算法如 FOIL [29]。

参 考 文 献

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (*SIGMOD*'93), pp. 207-216, May1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Databases* (*VLDB*'94), pp. 487-499, Sep. 1994.

[3] A. Asuncion and D. J. Newman. UCI Machine Learning Repository, 2007. http://www.ics.uci.edu/~mlearn/MLRepository.html. University of California, Irvine, School of Information and Computer Sciences.

[4] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman & Hall/CRC, Jan. 1984.

[5] C. E. Brodely and P. E. Utgoff. Multivariate Decision Trees. *Machine Learning*, 19:45-77, 1995.

[6] P. Clark and T. Niblett. The CN2 Induction Algorithm. *Machine Learning*, 3(4):261-283, 1999.

[7] W. Cohen. Fast Efficient Rule Induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 115-123, 1995.

[8] T. G. Dietterich. An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Machine Learning*, 40(2):139-157, 2000.

[9] S. Dzeroski and N. Lavrac, eds. *Relational Data Mining*. Springer, Berlin, 2001.

[10] U. M. Fayyad and K. B. Irani. On the Handling of Continuous-Valued Attributesin Decision Tree Generation. *Machine Learning*, 8(1):87-102, Jan. 1992.

[11] Y. Freund and L. Mason. The Alternating Decision Tree Learning Algorithm. In *Proceedings of the Sixteenth International Conference on Machine Learning* (*ICML* 1999), pp. 124-133, 1999.

[12] Y. Freund and R. E. Schapire. A Short Introduction to Boosting. *Journal of the Japanese Society for Artificial Intelligence*, 14(5):771-780, Sep. 1999.

[13] J. H. Friedman. A Recursive Partitioning Decision Rule for Nonparametric Classification. *IEEE Transactions on Computers*, 26(4):404-408, Apr. 1977.

[14] J. Gehrke, V. Ganti, R. Ramakrishnan, and W.-H. Loh. BOAT: Optimistic Decision Tree

Construction. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (*SIGMOD*'99), pp. 169-180, 1999.

[15] J. Gehrke, R. Ramakrishnan, and V. Ganti. RainForest: A Framework for Fast Decision Tree Construction of Large Datasets. *Data Mining and Knowledge Discovery*, 4(2/3):127-162, 2000.

[16] E. B. Hunt, J. Marin, and P. J. Stone. *Experiments in Induction*. Academic Press, New York, 1966.

[17] R. Kohavi, D. Sommerfield, and J. Dougherty. Data Mining Using MLC++: A Machine Learning Library in C++. In *Proceedings of the Eighth International Conference on Tools with Artificial Intelligence* (*ICTAI*'96), pp. 234-245, 1996.

[18] D. Koller and M. Sahami. Toward Optimal Feature Selection. In *Proceedings of the Thirteenth International Conference on Machine Learning* (*ICML*'96),pp. 284-292, 1996.

[19] D. Kumar, N. Ramakrishnan, R. F. Helm, and M. Potts. Algorithms for Storytelling. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD*'06), pp. 604-610,Aug. 2006.

[20] B. Liu, W. Hsu, and Y. Ma. Integrating Classification and Association Rule Mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining* (*KDD*'98), pp. 80-86, Aug. 1998.

[21] M. Mehta, R. Agrawal, and J. Rissanen. SLIQ: A Fast Scalable Classifier for Data Mining. In *Proceedings of the 5th International Conference on Extending Database Technology* (*EDBT*'96), pp. 18-32, Mar. 1996.

[22] S. K. Murthy, S. Kasif, and S. Salzberg. A System for Induction of Oblique Decision Trees. *Journal of Artificial Intelligence Research*, 2:1-32, 1994.

[23] D. W. Opitz and R. Maclin. Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research*, 11:169-198, 1999.

[24] L. Parida and N. Ramakrishnan. Redescription Mining: Structure Theory and Algorithms. In *Proceedings of the Twentieth National Conference on Artificial Intelligence* (*AAAI*'05), pp. 837-844, July 2005.

[25] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1(1):81-106,1986.

[26] J. R. Quinlan. Simplifying Decision Trees. Technical Report 930, MIT AI Lab Memo, Dec. 1986.

[27] J. R. Quinlan. Decision Trees as Probabilistic Classifiers. In P. Langley, ed. ,*Proceedings of the Fourth International Workshop on Machine Learning*. Morgan Kaufmann, CA, 1987.

[28] J. R. Quinlan. Unknown Attribute Values in Induction. Technical report, Basser Department of Computer Science, University of Sydney, 1989.

[29] J. R. Quinlan. Learning Logical Definitions from Relations. *Machine Learning*,5:239-266, 1990.

[30] J. R. Quinlan. C4.5: *Programs for Machine Learning*. Morgan Kaufmann, 1993.

[31] J. R. Quinlan. Improved Use of Continuous Attributes in C4.5. *Journal of Artificial Intelligence Research*, 4:77-90, 1996.

[32] N. Ramakrishnan, D. Kumar, B. Mishra, M. Potts, and R. F. Helm. Turning CARTwheels: An Alternating Algorithm for Mining Redescriptions. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD*'04), pp. 266-275, Aug. 2004.

[33] R. Rastogi and K. Shim. PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning. In *Proceedings of the 24th International Conference on Very Large Data Bases* (*VLDB*'98), pp.

404-415, Aug. 1998.

[34] J. C. Shafer, R. Agrawal, and M. Mehta. SPRINT: AScalable Parallel Classifier for Data Mining. In *Proceedings of the 22th International Conference on Very Large Data Bases* (*VLDB*'96), pp. 544-555, Sep. 1996.

[35] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[36] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 Algorithms in Data Mining. *Knowledge and Information Systems*, 14: 1-37, 2008.

[37] L. Zhao, M. Zaki, and N. Ramakrishnan. BLOSOM: A Framework for Mining Arbitrary Boolean Expressions over Attribute Sets. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD*'06), pp. 827-832, Aug. 2006.

第 2 章　k-means

Joydeep Ghosh，Alexander Liu

2.1　引言
2.2　算法描述
2.3　可用软件
2.4　示例
2.5　高级主题
2.6　小结
2.7　习题
参考文献

2.1 引 言

本章介绍一种被广泛使用的直接聚类算法 k-means 算法。给定一个对象(或记录)的集合,所谓聚类(或拆分)就是把这些对象划分为多个组或者"聚簇",从而使得同组内对象间比较相似而不同组对象间差异比较大。换言之,聚类算法就是要将相似的对象放入同一个聚簇,而将不相似的对象分到不同的聚簇中。需要注意的是,在回归、分类等有监督学习任务中要定义类别标签或者目标值,但聚类过程的输入对象没有与之关联的目标信息(即类别标签或者目标值)。正因为如此,聚类通常归于无监督学习任务。由于无监督算法不需要带标签数据,所以适用于许多难以获取带标签数据的应用。在进行有监督学习任务之前,经常需要先利用聚类等无监督学习来探查数据集并挖掘其特性。由于聚类不使用类别标签,所以相似性的概念要基于对象的属性进行定义。应用不同则相似性的定义和聚类算法都会不同。所以,不同的聚类算法适用的数据集类型和挖掘目的都不一样。因此,"最优"聚类算法实际上依赖于具体的应用。所以,对于具体问题往往需要先多试几种不同的聚类算法,再从中挑出合适的。

k-means 算法是一种简单的迭代型聚类算法,它将一个给定的数据集分为用户指定的 k 个聚簇。实现和运行该算法都很简单,它的速度也比较快,同时又易于修改,所以在实际中使用非常广泛。它可以说是数据挖掘领域发展史中最为重要的算法之一。

历史上,许多不同学科领域的研究人员都对基本的 k-means 算法进行过研究,其中知名的主要有 Lloyd(1957, 1982)[16], Forgey(1965)[9], Friedman 与 Rubin(1967)[10], McQueen(1967)[17]等人。Jain 与 Dubes[13]详细描述了 k-means 算法的发展历史和多种变体。Gray 与 Neuhoff[11]从爬山算法这个大背景出发,对 k-means 算法进行了非常精彩的论述。

在接下来的内容中,我们将描述 k-means 算法的细节并讨论其局限,还会给出 k-means 算法处理几个人工数据集和实际数据集的例子。此外,我们还将简单讨论一些 k-means 算法扩展。需要说明的是,我们所列出的 k-means 算法扩展远不完备,读者可根据自己的关注点更深入地去研究和探索 k-means 算法的方方面面。

2.2 算法描述

k-means 算法的输入对象是 d 维向量空间中的一些点。因此,它是对一个 d 维向量的点集 $D=\{x_i \mid i=1,\cdots,N\}$ 进行聚类,其中 $x_i \in \mathcal{R}^d$ 表示第 i 个对象(或称"数据点")。k-means 聚类算法会将集合 D 划分成 k 个聚簇。也就是说,k-means 算法对 D 中所有的数据点进行聚类处理,将每个点 x_i 都归于且仅归于 k 个聚簇中的一个。我们可以为每一个点分配一个聚簇标识以记录该点分配到哪一个聚簇中去了。拥有相同聚簇标识的点属于同一个聚簇,拥有不同聚簇标识的点属于不同的聚簇。我们可以定义一个长度为 N 的聚簇成员向量 m,其分量 m_i 表示点 x_i 的聚簇标识。

k 值是基本 k-means 算法的一个关键的输入。确定 k 值的典型做法是依据某些先验知

识，例如，集合 D 中实际存在的或当前应用所预期的聚簇数量，当然也可以通过测试不同的 k 值进行探查聚簇的类型信息，从而最终选定合适的 k 值。我们将会在后面的章节中讨论在 k 值没有被预先指定时如何来选取。其实，如何选取 k 值对理解 k-means 算法如何对数据集 D 进行划分没有关系。

在 k-means 算法中，每个聚簇都用 $\mathcal{R}^d$ 中的一个点来代表。可将这些聚簇用集合 $C=\{c_j|j=1,\cdots,k\}$ 来表示。这 k 个聚簇代表有时也被称为聚簇均值或者聚簇中心，在介绍了 k-means 算法的目标函数之后，大家就会明白为什么会有这样的名称。

聚类算法通常基于“紧密度”或者“相似度”等概念对点集进行分组。具体到 k-means 算法，默认的紧密度度量标准是欧几里得距离。k-means 算法实质是要最小化一个如下的非负代价函数：

$$\text{Cost}=\sum_{i=1}^{N}(\text{argmin}_j\ ||\ x_i-c_j\ ||_2^2) \tag{2.1}$$

换言之，k-means 算法要最小化目标是：每个点 x_i 和离它最近的聚簇代表 c_j 之间的欧几里得距离的平方和。式(2.1)通常被称为 k-means 目标函数。

算法 2.1 对描述了 k-means 通过迭代的方式对点集 D 进行聚类，该迭代过程主要包括了交替执行的两个步骤：(1)重新确定点集 D 中每个数据点的聚簇标识；(2)基于每个聚簇内所有数据点算出新的对应的聚簇代表。完整的算法流程包括，首先，用 $\mathcal{R}^d$ 中的 k 个数据点作为初始的聚簇代表。挑选这些初始种子的方法可以是从数据集中随机抽取样本，或者在数据集的某个子集上聚类得到聚簇代表，抑或取数据全局均值的 k 次扰动等。在算法 2.1 中，我们是通过随机选取 k 个点来启动算法。然后，该算法依次执行下面两个步骤的迭代直至收敛。

步骤 1：再分数据。将每个数据点分配到当前与之最近的那个聚簇中心，同时打破了上次迭代确定的归属关系。这一步会对全部数据进行一个新的划分。

步骤 2：重定均值。重新确定每一个聚簇代表，即计算所有分配给该聚簇代表的数据的中心(如算数平均值)。这样做的合理性是基于如下的考量：对于一个给定的点集，为了最小化所有点与代表之间的欧几里得距离平方和这个代价，代表的计算方式就应是这些点的均值。这就是为什么聚簇代表经常又被称为聚簇均值或者聚簇中心的原因，其实这也是 k-means 算法的这个名字的由来。

当 $C=\{c_j|j=1,\cdots,k\}$ 不再变化时，算法收敛。对于式(2.1)中定义的 k-means 目标函数，可以证明，只要上述第 2 步得到的新的“均值”同上一次迭代比还有变化，那该目标函数的值就仍会降低。进一步，还可以证明在有限步的迭代后该算法必定收敛。

注意到每一次迭代都需要 $N\times k$ 次比较，这也就决定了每次迭代的时间复杂度。k-means 算法收敛所需的迭代次数可能依赖于 N，直觉上应该与数据集的大小成线性关系，当然根据不同具体条件会有所变化。同时，由于比较操作关于 d 也是线性的，所以 k-means 算法的复杂度关于数据维度也是线性的。

算法 2.1 k-means 算法

输入：数据集 D，聚簇数 k
输出：聚簇代表集合 C，聚簇成员向量 **m**

/ * 初始化聚簇代表 C * /
从数据集 D 中随机挑选 k 个数据点
使用这 k 个数据点构成初始聚簇代表集合 C
repeat
　/ * 再分数据 * /
　将 D 中的每个数据点重新分配至与之最近的聚簇均值
　更新 **m**(m_i 表示 D 中第 i 个点的聚簇标识)
　/ * 重定均值 * /
　更新 C(c_j 表示第 j 个聚簇均值)
until 目标函数 $\sum_{i=1}^{N}(\text{argmin}_j \| x_i - c_j \|_2^2)$ 收敛

局限性——k-means 算法本质上是一种面向非凸代价函数优化的贪婪下降求解算法，所以仅能获得局部最优解。此外，该算法还对初始聚簇中心的位置非常敏感，也就是说，即便是同一数据集，如果聚簇代表集合 C 的初始化不同，最终获得的聚簇可能会差异很大。一个糟糕的初始化聚簇代表集合会导致一个糟糕的最终聚簇集合。在 2.3 节中，我们将用人工数据集和实际数据集的相关例子来说明这个问题。为了解决局部最小点问题，我们可以基于不同的初始聚簇中心多次运行该算法，从中挑选最好的结果，或者，对收敛解进行受限的局部搜索也是可行的。还有很多其他办法，例如文献[14]介绍了一些防止 k-means 收敛到局部最小的措施。文献[8]介绍了 k-means 的一系列不同的初始化方法，此外还包括该算法在其他方面的一些局限性的讨论。

前面提到，选择最优的 k 值是比较困难的。如果我们有一些关于数据集的先验知识，譬如知道数据集可以分成多少个部分，那么自然可以将 k 指定为这个数量。否则，我们必须使用其他准则来选择 k，这就是模型选择问题。最容易想到的一个朴素的办法是尝试多个不同的 k 值，并选择使 k-means 目标函数式(2.1)最小化的那个 k 值。但稍加分析就发现此路不通，目标函数的值对 k 值选择的问题并不适用。比如，最优解的代价会随着 k 的增长而降低，当聚簇的个数增长到和数据点的个数一样时代价降为 0。这就意味着不宜使用该目标函数进行以下工作：

(1) 直接比较具有不同聚簇数量的聚类结果。

(2) 寻找最优 k 值。

所以，如果事先无法知道理想的 k 值，人们一般会尝试取多个不同的 k 值多次运行 k-means，然后使用一些别的更合适的准则从结果中挑选一个。譬如，SAS 使用了立方聚类准则，就是在 k-means 的原代价函数式(2.1)上再加上一个复杂性控制项(其值会随着 k 的增大而增大)，通过最小化这一经过调整的代价函数来确定合适的 k 值[20]。又或者，可以在一个恰当的停止准则的辅助下逐渐增加聚簇的数目。有一种名为 bisecting k-means[21] 等聚类算法，它在初始时将所有数据当成一个聚簇，然后递归地将最不紧凑的聚簇用 2-means 拆分为两个聚簇，直至满意。有一种用于向量量化的著名算法 LBG[11]，它每次将聚簇的数量翻番直至获得一个大小合适的代码本。这两种方法一定程度上降低和减轻了 k-means 需要求事先确定 k 值的要求。此外，许多其他学者也研究过这个问题，如文献[18]和[12]的工作。

除了上面提到的这些局限，还有几个 k-means 算法难以处理问题。为了便于理解，设想

我们用一个具有均一和各向同性协方差矩阵($\Sigma=\sigma^2 I$)的 k 维混合高斯分布处理数据拟合问题(此处 I 表示单位矩阵),这样就得到一个"柔性"k-means 聚类算法。更准确地说,这种方法会将每个数据点"柔性分配"(即以一定概率)给该模型的各个混合分量,如果要求"刚性分配"(即每个数据点仅能被分配给那个与之最相似的分量[3])我们就又回到了原来的标准 k-means 算法了。从这种联系中,可以非常清楚地看到 k-means 算法内在地假设数据集是由 k 个球(或超球)分量混合而成的,每一个聚簇对应一个分量。从这一隐含假设就可以推出,如果实际的数据集并不是若干球形高斯分布的重叠,则 k-means 就会不稳定。比如数据集中存在非凸形状的聚簇时,k-means 就不能很好工作。为了应对这个问题,可以在执行聚类前先用"白化"等方法对数据进行缩放,或者选择与数据集更匹配的距离度量,譬如,基于信息论的聚类算法通常会将数据点用离散概率分布表示,进而使用 KL 距离对两个分布进行度量,即可获得两个数据点之间的距离了。近期的一些研究表明,如果我们在 k-means 的"再分数据"这一步骤中,从被统称为 Bregman divergences 的一大类偏差度量中任选一种作为数据点间的距离度量(除此外不对算法作其他更改),则 k-means 算法仍能保持确定收敛、线性可分边界,可扩展等本质属性[1]。这就是说,对于非常多种数据集,只要找到与之匹配的偏差度量方法,使用 k-means 就能获得良好的聚类效果。

另一种处理非凸聚类的方法是用别的算法与 k-means 形成配套。譬如,我们可以先用 k-means 将数据聚类成很多个组,然后再用单链层次聚类方法(single link hierarchical clustering)凝聚出更大的聚簇,这种方法有若干优点:首先,它比较利于发现检测出复杂的形状,然后,它可以减少聚类结果对于初始化的敏感性,最后,由于层次化方法自然支持多分辨率聚类结果,所以不用指定一个准确的 k 值,只需在生成初始聚簇时简单给一个较大的 k 值即可。

k-means 算法还存在对噪声点敏感的问题,从本质上讲均值并不是一种稳健统计量。在用 k-means 聚类前,通过预处理移除噪声点往往非常有用的,同样,在聚类后,对聚类结果进行一些后处理效果也很好,如删除过小的聚簇,或将彼此接近的一些聚簇合并成一个更大的聚簇。Ball 与 Hall 在 1967 年研究的 ISODATA 算法就对 k-means 算法实施了有效的预处理和后处理过程。

k-means 算法还有一个潜在的问题是可能产生"空聚簇"[4]。特别是使用一个较大的 k 值并且/或者数据存在于高维空间中的情况下,在 k-means 执行过程中,有可能在某个阶段会对于某个聚簇代表 c_j,会发生所有 D 中的点 x_i 都更接近其他聚簇代表不是 c_j 的情况,这样,把 D 中的每个数据点分配至与之最近的聚簇,那么第 j 个聚簇就一个数据点也分不到。换而言之,第 j 个聚簇为空。标准的 k-means 算法并不会特别处理空聚簇问题,我们可以采取一些简单有效的方法,如从最大的聚簇中"偷出"一些点来重新初始化空聚簇的聚簇代表。

2.3 可用软件

由于 k-means 算法的简单性、有效性和历史上的重要性,人们开发了各种形式的运行 k-means 算法的软件。该算法已经成为许多流行的数据挖掘软件包的一个标配。譬如,在 Weka、SAS 等软件中都有一个对应 k-means 算法的过程 FASTCLUS。还有,MATLAB 的

许多工具箱中都包含了 k-means 算法的实现。就连微软的 excel 软件都可以使用 k-means 算法(在 XMLMiner 软件包中实现)。当然,在互联网上有更多独立开发的 k-means 算法工具。

其实,直接来编程实现 k-means 算法非常容易,本文的读者可以尝试实现自己的 k-means 算法。

2.4　示　　例

我们先在人工数据集上演示一下 k-means 算法是如何工作。

我们从二维高斯分布采样生成包含 4 个聚簇(k=4)的人工数据集,如图 2.1 所示,每种颜色数据点对应一个高斯分布。其中,蓝色数据点有 200 个,采样自均值(−3,−3)、协方差矩阵 0.625×I 的高斯分布(I 为单位矩阵,下面同);绿色数据点有 200 个,采样自均值(3,−3)、协方差矩阵为 I 的高斯分布;最后,对两个高斯分布来采样,分布获得黄色和红色的数据点,由于这两个分布靠得很近,我们看到这两种颜色的数据点存在很大程度的重叠,其中黄色数据点有 150 个,对应高斯分布的均值为(−1, 2)、协方差矩阵为 I;其中红色数据点有 150 个,对应高斯分布的均值为(1, 2)、协方差矩阵为 I。尽管这个人工生成数据集中红色数据点和黄色数据数据点存在重叠,我们还是期望 k-means 算法能得到较好的聚类结果,因为此处已经知道了正确的 k 值,并且这些数据是通过混合球形高斯分布产生的,这些都很好地符合了该算法的隐含假设。

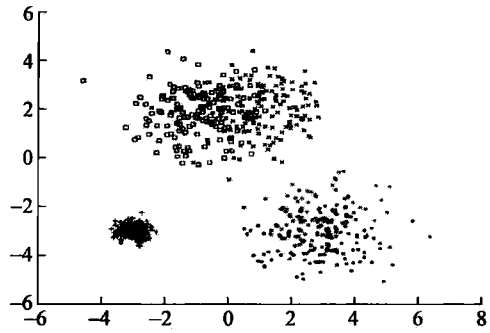


图 2.1　例子中采用的人工数据集:数据点采样自 4 个高斯分布的混合

k-means 算法的第一步是初始化聚簇代表,如图 2.2(a)所示,从数据集中随机选出 k 个点作为初始的聚簇代表。在这张图以及后续的图中,每个聚簇代表 C 都用一个带黑边的较大(比数据点大)的有色圆圈所表示。每种颜色对应一种聚簇标识,所以被分配至该聚簇的数据点也都用与聚簇标识相同的颜色表示。需要说明的是这些颜色与图 2.1 中的颜色并没有确定的关系。在图 2.2(a)中,由于数据点还没有被分配聚簇标识,所以它们暂时都用黑色表示。

k-means 算法的下一步是将每个数据点分配至距之最近的聚簇代表,如在图 2.2(b)所示,每个数据点都被着色以匹配自己的聚簇代表。k-means 算法的第三步,如图 2.2(c)所显

示，是更新 k 个聚簇的聚簇代表，具体方法是用当前所有被分配至该聚簇的数据点的均值来作新的聚簇代表。我们采用一个专门符号——黑“X”标记出老的簇代表，而使用一个带黑边的有色圆圈表示新的聚簇代表。同时，在老聚簇均值与新聚簇均值间连一条线，如此便于我们观察到聚簇代表的移动和每个聚簇的当前中心在何处。

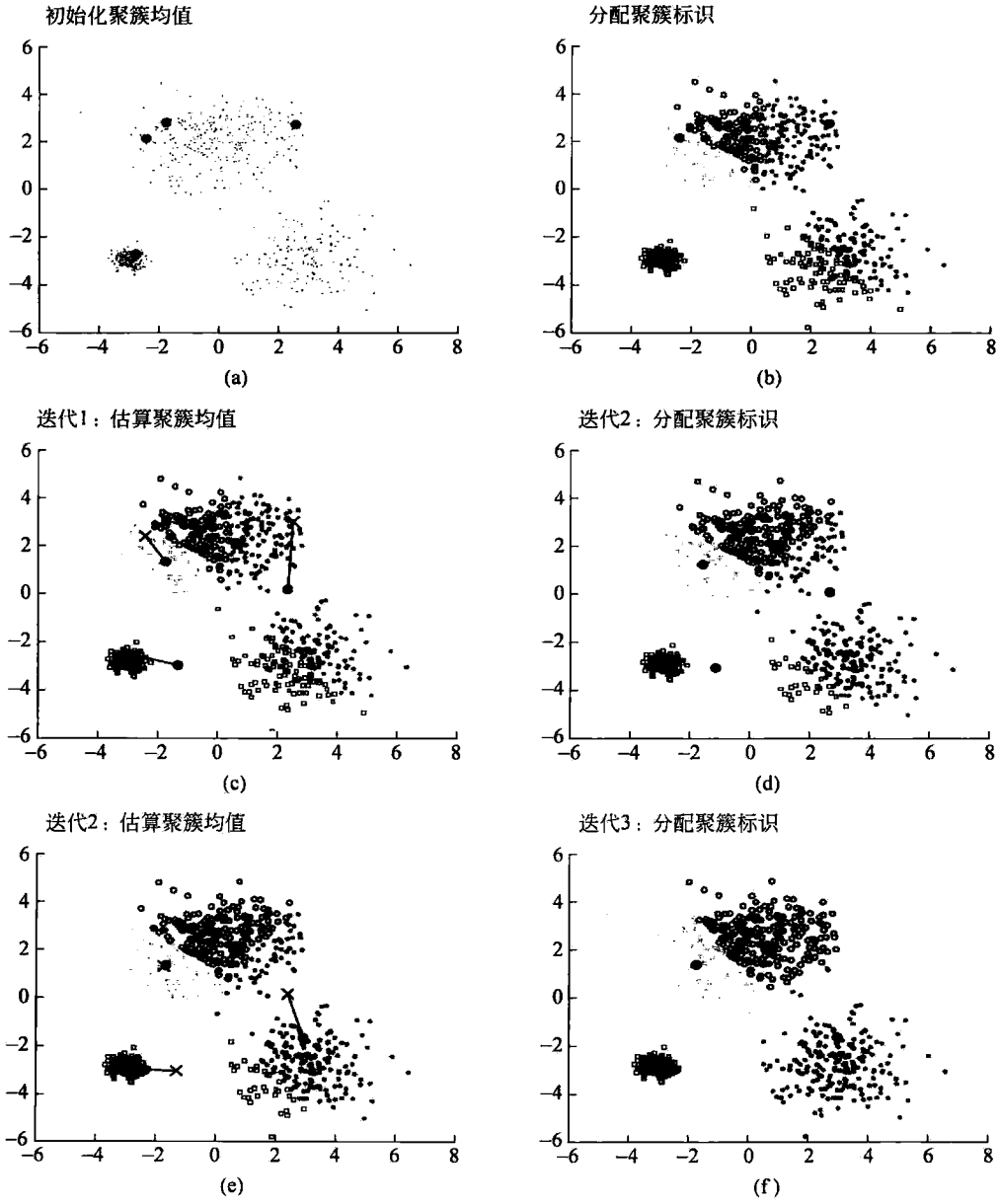


图 2.2 作用于人工数据集上的 k-means 算法

现在，k-means 算法就可以迭代执行两个步骤，即“分配 D 中的每个数据点至最近的聚簇代表，更新 k 个聚簇代表”，如此往复直至最终收敛。在图 2.2 和图 2.3 中展示了 k-means 算法的前 4 次迭代，最后收敛得到的聚簇在图 2.3(d)中展示。这个例子执行 8 次迭

代后最终收敛，后 4 次迭代聚类结果变化很小，所以为了节省空间就省略了这些步骤的图片。对比图 2.3(d)与图 2.1，k-means 算法找到的聚簇和实际的底层分布符合得非常好。

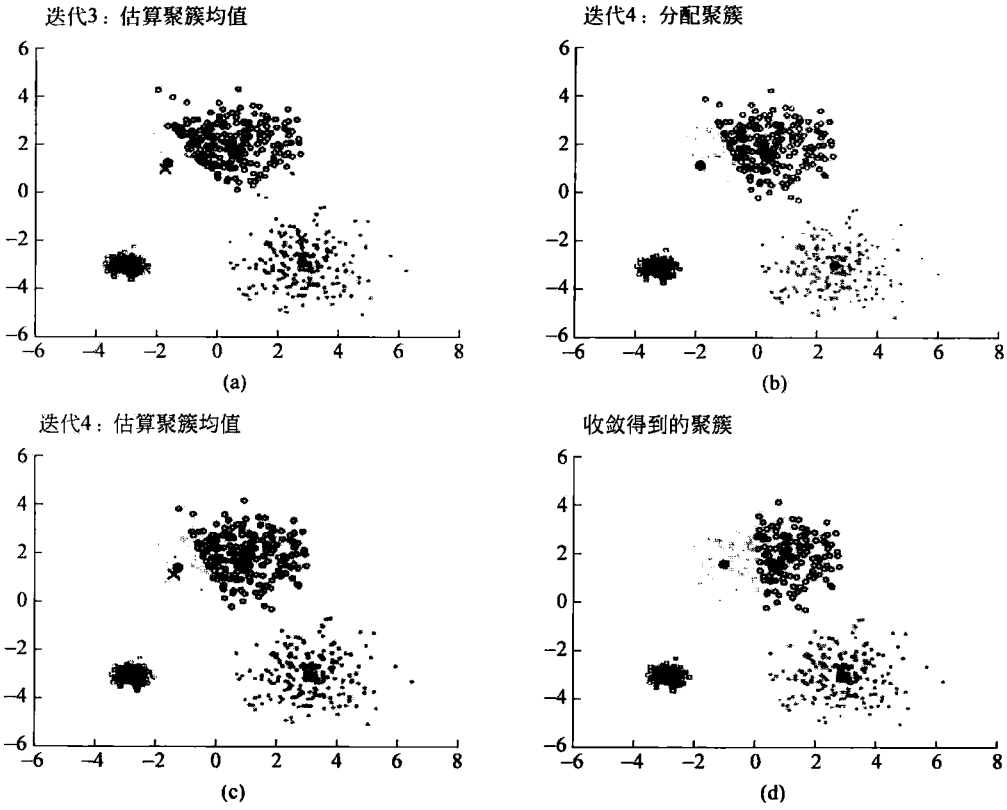


图 2.3　作用于人工数据集上的 k-means 算法(续图 2.2)

我们在前面提到过 k-means 算法对初始聚簇代表的选择是敏感的。现在我们选取不当的 k个初始簇代表(如图 2.4 所示)，然后看看算法的效果如何。所用的数据集和图 2.2 与图 2.3 是同一个。图 2.4(a)与图 2.4(c)用了两个不当的初始化，可以看到最终聚类效果比较差(如图 2.4(b)与图 2.4(d)所示)，该聚类结果没能很好地对应上数据真实的底层分布。

最后，我们将用一个简单而经典的数据集来验证一下 k-means 算法的性能。在本例中，我们使用 Iris 数据集(来自 UCI 数据挖掘资源库)，该数据集包含被分成 3 类的共 150 个数据点，每个类别代表鸢尾属植物的一个花种，每个类别包含 50 个数据点。数据本身共有 4 个维度(分别是花萼宽度、花萼长度、花瓣宽度和花瓣长度)，但实际只有两个维度(花瓣宽度与花瓣长度)对区分三个类别是有用的。在图 2.5(a)中，用花瓣宽度与花瓣长度两个维度显示了 Iris 数据集的情况。

在图 2.5(b)中，我们展示了 k=3 时用 k-means 算法对 Iris 数据集进行聚类的结果，此处只用了花瓣长度和花瓣宽度这两个维。结果显示 k-means 算法可以将数据点很好地聚类，我们看到每个聚簇中包含的花几乎都是同一个花种。

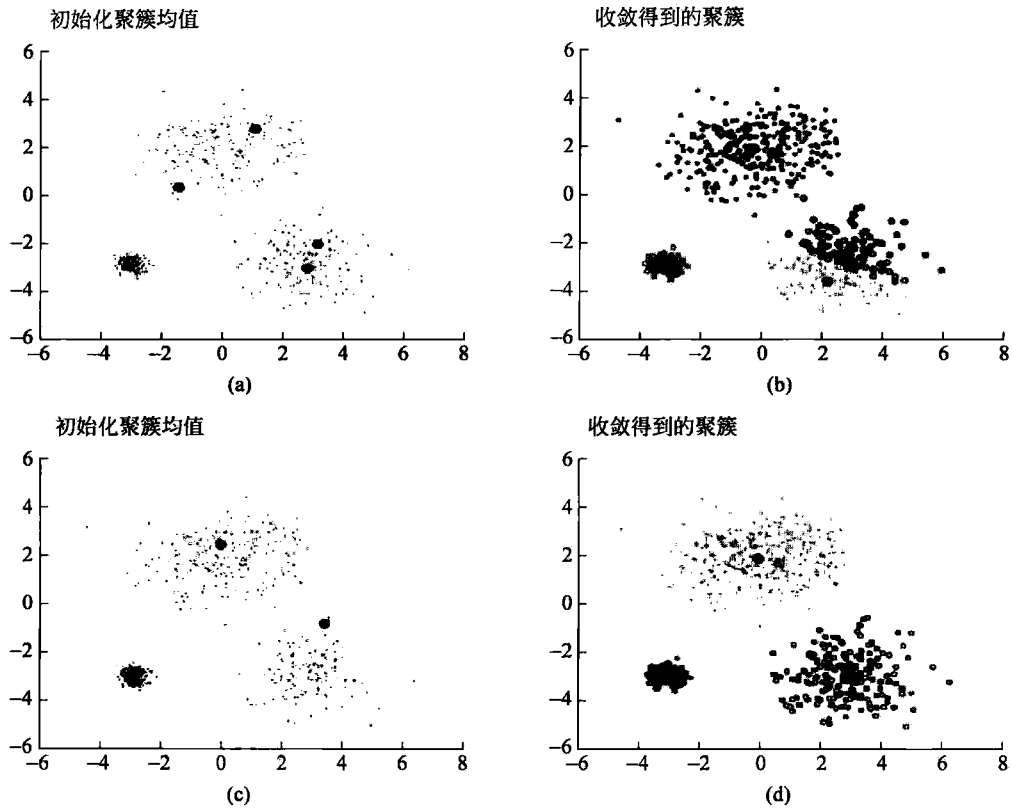


图 2.4 选取差的初始聚簇代表得到差的聚类结果的例子；这些结果不能很好地反应数据真实的底层分布

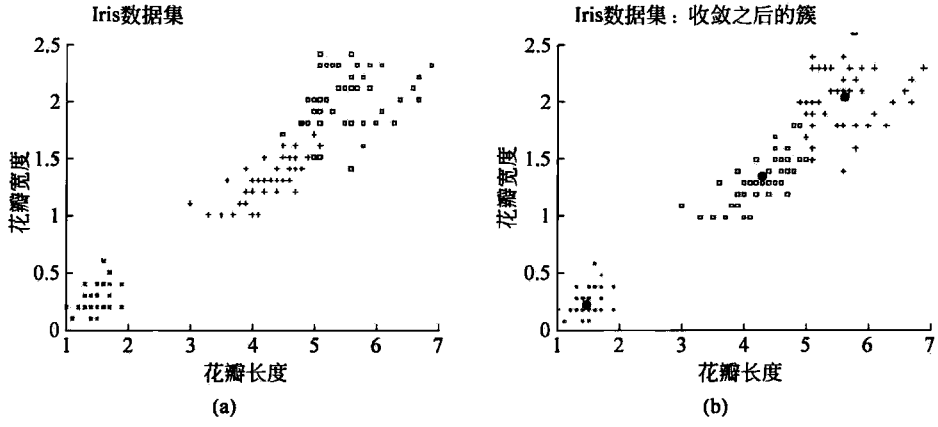


图 2.5 每种颜色代表一个聚簇，注意(a)和(b)中的颜色没有对应关系

(a) Iris 数据集：每种颜色表示一个花种；(b) 在 Iris 数据集上运用 k-means 聚类的结果

2.5　高级主题

本节我们将讨论一些对 k-means 算法的推广、扩展以及与其他算法的关联。过去对 k-means 的研究相当丰富，就是现在也十分活跃。所以，本节无法进行详尽的探讨，相反，我们的目标是对前面有关 k-means 的讨论进行一些必要的补充。

如前所述，标准的 k-means 算法同将基于 k 分量各向同性混合高斯分布的数据拟合问题密切关联，进而，将经典的距离度量扩大到所有 Bregman divergences 距离度量，就可以用 k 分量混合指数族分布进行数据拟合。另一个大的推广，就是把原本仅当做 $\mathcal{R}^d$ 中的点的“均值”看作概率模型。进而在 k-means 的“数据再分”这步中，每个数据点被分配给最有可能生成该数据点的那个模型；在“重定均值”的这步中，用分配好的数据集对模型参数进行最佳拟合，这种被称为基于模型的 k-means 算法[23]有处理复杂数据的能力，例如用隐马尔可夫模型描述的序列数据。

我们也可以实现“核”k-means 算法，即将 k-means 算法与核方法结合，这样的话，虽然聚簇边界在原空间是非线性的，但在核函数所隐含的高维空间中却可以变换成线性的，从而能处理复杂形状的聚簇。Dhillon 等人[5]分析了基于核方法的 k-means 算法同谱聚类方法之间存在紧密的关联。K-medoid[15]算法与 k-means 算法很相似，差别在于 K-medoid 算法要求重心(medoid)必须属于被聚类的数据集。模糊 c-means[6]的情形也是类似的，只不过该方法的聚簇隶属函数是模糊的，而不是确切的。

为了有效处理超大数据集，必须对 k-means 算法加速。人们在这方面已经进行了大量的努力，例如通过使用 kd-树[19]或利用三角不等式[7]，避免在“数据再分”这步计算所有＜数据点，中心＞对的距离。

最后，我们讨论两种对 k-means 算法的简单扩展。第一种扩展被称为柔性 k-means，在标准 k-means 算法中，每个点 x_i 属于且仅属于一个聚簇，但柔性 k-means 算法放松了这个约束，即每个点 x_i 依概率赋给每一个聚簇。柔性 k-means 算法中，每个点 x_i 都有一个 k 维概率(或权重)向量用来描述该点属于每个聚簇的可能性。这些权重是以点 x_i 到 C 中每个聚簇代表的距离为基础的，x_i 来自聚簇 j 的概率正比于 x_i 和 c_j 之间的相似性。这种情况下，聚簇代表是基于数据集 D 的所有点(而非一个聚簇的点)关于聚簇均值的期望的计算来得到。

第二种 k-means 的扩展与半监督学习有关。在引言部分，我们谈了有监督学习和无监督学习之间的区别。简单地讲，有监督学习需使用类别标签，而无监督学习则不用，而 k-means 算法正是一种纯粹的无监督的算法。除了这两种基本的学习类型外，还存在一类被称为半监督学习的类型，它能同时使用已标记和未标记的数据。可见，半监督学习结合了有监督方法和无监督方法的优势。有监督学习方法通常要求大量的已标记数据，当已标记数据很少时，半监督方法就能发挥作用了。无监督的学习方法虽然不用类别标签，但它学到的模型往往不符合我们面对的应用。在执行 k-means 算法过程中，我们实际上对最终聚簇没有控制机制，那这些聚簇就可能不对应(也可能对应)我们期望挖掘出的隐含概念。例如，在图 2.5(b)中，由于初始化选得不好，就造成最终得到的聚簇与 Iris 数据集中的花种不对应。而半监督方法借助于已标记数据点的指导，就更可能挖掘出同类别标签对应良好的聚簇。

对半监督 k-means 的研究可以参考文献[22]和[2]。文献[2]中提出的算法称为 seeded k-means，它比较简单，使用已标记的数据帮助初始化 k 值和聚簇代表 C。在这种方法中，k 值设定为已标记数据中的类别数，聚簇代表 C_j 根据第 j 类所有已标记点的均值来初始化。请注意，该算法与无监督 k-means 不同在于其第 j 个聚簇和第 j 个类别之间的对应关系是已知的。除了初始化方法的不同外，seeded k-means 算法与标准 k-means 算法是一样的，也是要执行两个步骤(再分数据和重定均值)的迭代直至收敛。

2.6 小　　结

k-means 算法使用简单的迭代将数据集聚成 k 个类，迭代的核心步骤有两个：(1)为数据集中的每个点寻找最近的聚簇代表，并由此将数据集聚类；(2)重新估算每个聚簇代表。其不足之处主要是该算法对初始化条件和 k 值选择敏感。

尽管如此，k-means 仍然是实践中使用最广泛的聚类算法。该算法具有简单、易懂、良好的可伸缩性等显著优点，而且对其稍加修改就能应用于半监督学习或者流数据处理等多种不同应用场景。长期以来，人们一直坚持改进和扩展 k-means 算法，使得该算法充满了活力。

2.7 习　　题

1. 请使用标准测试基 Iris 数据集(从 UCI 在线数据挖掘仓库获取)运行 k-means 算法来获取图 2.5(b)所示结果。实验仅需使用属性“petal width”和属性“petal length”。

请观察：k 值不取 3 时结果如何？不同的聚类初始化如何影响最终结果？为什么这些结果与图 2.5(b)给出的结果差异很大？

2. 请证明 k-means 目标函数的值会收敛。

3. 请描述 k-means 算法相对于其他聚类算法(比如 agglomerative clustering)的 3 个优点和 3 个缺点。

4. 请描述或画出一个 k-means 算法难以处理的二维示例。

5. k-means 算法的聚簇均值收敛后，聚簇边界的形状如何？它与 Voronoi tesselations 有什么关系？

6. k-means 算法能否保证聚簇内的点比聚簇间的点更相似？请证明或证伪：k-means 算法收敛后，属于同一聚簇的两点间的欧氏距离总是比分属不同聚簇的两个点间的欧式距离小。

7. 假设数据集 D 有 10 个点，让 k-means 算法在该数据集上运行两次。用向量 m 表示聚簇标识(向量 m 的第 i 个分量 m_i 表示 D 的第 i 个点的聚簇标识)。

设首次运行得到的聚簇标识是 $m^1=[1, 1, 1, 2, 2, 2, 3, 3, 3, 3]$，二次运行得到的聚簇标识是 $m^2=[3, 3, 3, 1, 1, 1, 2, 2, 2, 2]$。

请问两者之间有何差异？聚簇标识是什么意思？如何比较不同的聚类结果？知道类别标签的情况下又如何比较聚类结果？

8. 请亲自动手实现 k-means 算法和从 k-Gaussian 分布采样生成数据集，再在这个人工数据集上测试你的 k-means 实现并看看算法要多少次迭代才收敛。

9. 基于习题 8 的算法实现，画出取不同 k 值时每个点到其聚簇均值的距离的平均值。这个距离的平均值是否是确定 k 值的一个好方法？为什么？当 k 值等于数据集的样本数量时情形又当如何？

10. 请从您自己的兴趣出发来研究和描述标准 k-means 算法的某一种扩展，比如可以考虑改进计算效率、用于半监督学习、引入其他距离度量或者其他任何您能想到的思路。

参 考 文 献

[1] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. "Clustering with Bregman divergences", *Journal of Machine Learning Research* (*JMLR*), vol. 6, pp. 1705-1749, 2005.

[2] S. Basu, A. Banerjee, and R. Mooney. "Semi-supervised clustering by seeding", *International Conference on Machine Learning* 2002, pp. 27 34, 2002.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning* (*Information Science and Statistics*). 2006.

[4] P. S. Bradley, K. P. Bennett, and A. Demiriz. "Constrained k-means clustering", *Technical Report MSR-TR*-2000-65, 2000.

[5] I. S. Dhillon, Y. Guan, and B. Kulis. "Kernel k-means: Spectral clustering and normalized cuts", *KDD* 2004, pp. 551 556, 2004.

[6] J. C. Dunn. "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters", *Journal of Cybernetics*, vol. 3, pp. 32 57,1974.

[7] C. Elkan. "Using the triangle inequality to accelerate k-means", *International Conference on Machine Learning* 2003, pp. 147 153, 2003.

[8] C. Elkan. "Clustering with k-means: Faster, smarter, cheaper", *Keynote talk at Workshop on Clustering High-Dimensional Data*, *SIAM International Conference on Data Mining*, 2004.

[9] E. Forgey. "Cluster analysis of multivariate data: Efficiency vs. interpretability of classification", *Biometrics*, 21, pp. 768, 1965.

[10] H. P. Friedman and J. Rubin. "On some invariant criteria for grouping data", *Journal of American Statistical Association*, 62, pp. 1159 1178, 1967.

[11] R. M. Gray and D. L. Neuhoff. "Quantization", *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2325 2384, 1998.

[12] G. Hamerly and C. Elkan. "Learning the k in k-means", *Neural Information Processing Systems*, 2003.

[13] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*, Prentice Hall,1988.

[14] T. Kanungo, D. M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu. "A local search approximation algorithm for k-means clustering", *Computational Geometry: Theory and Applications*, 28 (2004), pp. 89 112, 2004.

[15] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*, 1990.

[16] S. P. Lloyd. "Least squares quantization in PCM", unpublished Bell Lab. Tech. Note, portions

presented at the Institute of Mathematical Statistics Meet., Atlantic City, NJ, Sept. 1957. Also, *IEEE Trans. Inform. Theory* (Special Issueon Quantization), vol. IT-28, pp. 129 137, Mar. 1982.

[17] J. McQueen. "Some methods for classification and analysis of mutivariate observations", *Proc. 5th Berkeley Symp. Math., Statistics and Probability*, 1,pp. 281 296, 1967.

[18] G. W. Milligan. "Clustering validation: Results and implications for appliedanalyses", *Clustering and Classification*, P. Arabie, L. J. Hubery, and G. De Soete, ed., pp. 341 375, 1996.

[19] D. Pelleg and A. Moore. "Accelerating exact k-means algorithms with geometric reasoning", *KDD* 1999, pp. 227 281, 1999.

[20] D. Pelleg and A. Moore. "X-means: Extending k-means with efficient estimation of the number of clusters", *International Conference on Machine Learning* 2000, pp. 727 734, 2000.

[21] M. Steinbach, G. Karypis, andV. Kumar. "Acomparison of document clustering techniques", *Proc. KDD Workshop on Text Mining*, 2000.

[22] K. Wagstaff, C. Cardie, S. Rogers, S. Schr odl. "Constrained k-means clustering with background knowledge", *International Conference on Machine Learning* 2001, pp. 577-584, 2001.

[23] S. Zhong and J. Ghosh. "A unified framework for model-based clustering", *Journal of Machine Learning Research* (*JMLR*), vol. 4, pp. 1001-1037, 2003.

第 3 章　SVM：支持向量机

Hui Xue，Qiang Yang，Songcan Chen

3.1　支持向量分类器
3.2　支持向量分类器的软间隔优化
3.3　核技巧
3.4　理论基础
3.5　支持向量回归器
3.6　软件实现
3.7　当前和未来的研究
　　3.7.1　计算效率
　　3.7.2　核的选择
　　3.7.3　泛化分析
　　3.7.4　结构化支持向量机的学习
3.8　习题
参考文献

支持向量机(SVM)是在所有知名的数据挖掘算法中最健壮、最准确的方法之一，主要包括支持向量分类(SVC)和支持向量的回归器(SVR)。SVM 最初是由 Vapnik [1-11] 在 20 世纪 90 年代发展而来的，该方法建立在统计学习理论的坚实基础上。SVM 可以从大量训练数据中选出很少的一部分用于模型构建，而且通常维数不敏感。在过去的十年中，SVM 在理论和实践两方面的发展都非常快。

3.1 支持向量分类器

对于两类线性可分学习任务，SVC 要找到一个间隔最大的超平面将两类样本分开，最大间隔能确保该超平面具有最好的泛化能力。泛化能力是指不但要在训练数据有好的分类性能(例如精度等)，也要能对于和训练数据具有同一分布的新数据进行高精度的预测。

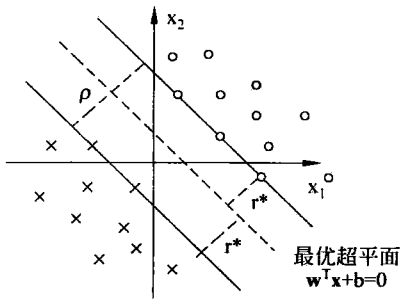


图 3.1 线性可分的情况下在 SVC 的最优超平面

从直观上看，间隔就是两个类之间的空间大小或者超平面所确定分离程度。从几何上讲，间隔对应于数据点到超平面的最短距离。图 3.1 说明了一个二维输入空间上的最优超平面的几何结构。令 $\mathbf{w}$ 和 b 分别表示权重向量和最优超平面偏移，则相应的超平面可以被定义为：

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}+\mathrm{b}=0 \tag{3.1}$$

样本 $\mathbf{x}$ 到最优超平面的几何距离[12,13]是：

$$\mathrm{r}=\frac{\mathrm{g}(\mathbf{x})}{||\mathbf{w}||} \tag{3.2}$$

其中 $\mathrm{g}(\mathbf{x})=\mathbf{w}^{\mathrm{T}}\mathbf{x}+\mathrm{b}$ 是超平面确定的判别函数[7]，也称为给定 $\mathbf{w}$ 和 b 的 $\mathbf{x}$ 的泛函间隔。

SVC 就是要寻找确定最优超平面的参数值 $\mathbf{w}$ 和 b，以最大化两个类间的分离间隔(式(3.5)中的 ρ)，该量取决于两个类之间的最短几何距离 r^*。因此，SVC 也被称为最大间隔分类器。不失一般性，我们将泛函间隔[7]固定为 1，那么对给定训练集 $\{\mathbf{x}_i,\mathrm{y}_i\}_{i=1}^{n}\in\mathbf{R}^m\times\{\pm1\}$，有：

$$\begin{cases}\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+\mathrm{b}\geqslant 1 & \text{当 } \mathrm{y}_i=+1\\ \mathbf{w}^{\mathrm{T}}\mathbf{x}_i+\mathrm{b}\leqslant -1 & \text{当 } \mathrm{y}_i=-1\end{cases} \tag{3.3}$$

有一些数据点 $(\mathbf{x}_i,\mathrm{y}_i)$ 使式(3.3)的等式成立，它们就是离最优超平面最近的数据点[13]，从而被称为支持向量。那么，支持向量 $\mathbf{x}^*$ 到最优超平面的几何距离就是：

$$\mathrm{r}^*=\frac{\mathrm{g}(\mathbf{x}^*)}{||\mathbf{w}||}=\begin{cases}\dfrac{1}{||\mathbf{w}||} & \text{当 } \mathrm{y}^*=+1\\ -\dfrac{1}{||\mathbf{w}||} & \text{当 } \mathrm{y}^*=-1\end{cases} \tag{3.4}$$

在图 3.1 中，分离的间隔 ρ 可以表示成

$$\rho=2\mathrm{r}^*=\frac{2}{||\mathbf{w}||} \tag{3.5}$$

为了找到间隔最大的超平面，SVC 要用 $\mathbf{w}$ 和 b 最大化 ρ：

$$\max_{w,b} \frac{2}{||w||}$$
$$y_i(w^T x_i + b) \geqslant 1, \quad (i = 1, \cdots, n) \tag{3.6}$$

等价于：

$$\min_{w,b} \frac{1}{2} ||w||^2$$
$$y_i(w^T x_i + b) \geqslant 1, \quad (i = 1, \cdots, n) \tag{3.7}$$

我们使用$||w||^2$而不是$||w||$是为了方便后续的优化过程。一般情况下，我们用拉格朗日乘数法来求解式(3.7)所示的约束优化问题(称为原问题)：

$$L(w,b,\alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i [y_i(w^T x_i + b) - 1] \tag{3.8}$$

其中 α_i 是对应第 i 个不等式的拉格朗日乘数。

对 $L(w,b,\alpha)$的 w 和 b 求偏导并使之为零，就可得到最优化条件：

$$\begin{cases} \dfrac{\partial L(w,b,\alpha)}{\partial w} = 0 \\ \dfrac{\partial L(w,b,\alpha)}{\partial b} = 0 \end{cases} \tag{3.9}$$

然后，可以得出：

$$\begin{cases} w = \sum_{i=1}^{n} \alpha_i y_i x_i \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases} \tag{3.10}$$

将式(3.10)代入拉格朗日函数式(3.8)，可以得到相应的对偶问题：

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$
$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{3.11}$$
$$\alpha_i \geqslant 0, \quad (i = 1, \cdots, n)$$

同时，再补充上 Karush-Kuhn-Tucker 条件：

$$\alpha_i [y_i(w^T x_i + b) - 1] = 0, \quad (i = 1, \cdots, n) \tag{3.12}$$

这里只有距离最优超平面最近的那些支持向量(x_i, y_i)对应的 α_i 非零，而其他所有 α_i 都等于零。

式(3.11)中的对偶问题是一个典型的凸二次规划问题。大多数情况下，采用序贯最小优化(SMO)算法[7]等适当的优化技术可以高效地求得全局最优解。

在确定最优拉格朗日乘子 α_i^* 后，我们可以计算式(3.10)中的最优权重向量 w^*：

$$w^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i \tag{3.13}$$

然后，用一个正的支持向量 x_s，可以算出最优偏置 b^*[13]：

$$b^* = 1 - w^{*T} x_s, \quad 当 \quad y_s = +1 \tag{3.14}$$

3.2 支持向量分类器的软间隔优化

最大间隔的 SVC 以及后面的 SVR 只是 SVM 算法的起点。要知道现实世界的许多问题并不是都是线性可分的，尤其是存在许多复杂的非线性可分的情形。如果样本不能被完全线性分开，那么情况就是：间隔为负，原问题的可行域为空，对偶问题的目标函数无限，这将导致相应的最优化问题不可解[7]。

要解决这些不可分问题，一般有两种方法。第一种是放宽过于严格的式(3.7)，构造出所谓的软间隔优化。另一种方法是运用核技巧将这些非线性问题线性化。在本节中，我们首先介绍软间隔优化。相对于软间隔 SVC，我们称源于式(3.7)优化问题的 SVC 为硬间隔 SVC。

想象一下，两个类有几个数据点混在一起，这些点对最大间隔超平面而言是训练错误。“软间隔”就是要扩展 SVC 算法以使超平面允许少量这样的噪声数据存在。具体说，就是引入松弛变量 ξ_i 来量化分类器的违规行为：

$$\begin{cases}\min\limits_{w,b}\dfrac{1}{2}\|\mathbf{w}\|^2+C\sum\limits_{i=1}^{n}\xi_i \\ y_i(\mathbf{w}^T\mathbf{x}_i+b)\geqslant 1-\xi_i,\quad \xi_i\geqslant 0,\quad (i=1,\cdots,n)\end{cases} \tag{3.15}$$

参数 C 用来平衡机器的复杂度和不可分数据点的数量，它可被视为一个由用户依据经验或分析选定的“正则化”参数。

松弛变量 ξ_i 的一个直接的几何解释是一个错分实例到超平面的距离。这个距离度量的是错分实例相对于理想的可分模式的偏差程度。使用与前面相同的拉格朗日乘子法，即可得出软间隔优化的对偶问题：

$$\begin{cases}\max\limits_{\alpha}W(\alpha)=\sum\limits_{i=1}^{n}\alpha_i-\dfrac{1}{2}\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \\ \sum\limits_{i=1}^{n}\alpha_i y_i=0,\quad 0\leqslant\alpha_i\leqslant C,\quad (i=1,\cdots,n)\end{cases} \tag{3.16}$$

比较式(3.11)与式(3.16)，可以发现松弛变量 ξ_i 没有出现在对偶问题里，线性可分与不可分的差异体现在约束 $\alpha_i\geqslant 0$(可分)被替换为更严格的约束 $0\leqslant\alpha_i\leqslant C$(不可分)。但是，其实这两种情况下非常相似的，比如权重向量 w 和偏移 b 的最优值的计算方法，以及关于支持向量的定义都是一致的[7,13]。

在不可分情况下，对应的 Karush-Kuhn-Tucker 补充条件是：

$$\alpha_i[y_i(\mathbf{w}^T\mathbf{x}_i+b)-1+\xi_i]=0,\quad (i=1,\cdots,n) \tag{3.17}$$

且

$$\gamma_i\xi_i=0,\quad (i=1,\cdots,n) \tag{3.18}$$

其中 γ_i 是对应于 i 的拉格朗日乘子，用于确保 ξ_i 非负[13]。在鞍点处，原问题的拉格朗日函数关于 ξ_i 的导数为零，通过求导得到：

$$\alpha_i+\gamma_i=C \tag{3.19}$$

结合式(3.18)和式(3.19)，有：

$$\xi_i=0\quad \text{if}\quad \alpha_i<C \tag{3.20}$$

接下来就得到最优权重 $\mathbf{w}^*$：

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i \tag{3.21}$$

选择训练集里的任一满足 $0 < \alpha_i^* < C$ 和相应的 $\xi_i = 0$ 数据点$(\mathbf{x}_i, y_i)$，代入式(3.17)[13]即可得到最优偏移量 b^*。

3.3 核 技 巧

核技巧是另一种用来处理线性不可分问题的方法。该方法要做的就是定义一个能恰当地计算给定数据对内积的核函数，其功能是将数据从输入空间非线性变换到特征空间，特征空间具有更高甚至无限的维度，从而使得数据在该空间中被转换成线性可分的。Cover 模式可分性定理表明，复杂的模式分类问题通过非线性映射嵌入到高维空间后比在低维空间中更可能被线性分开[13]。

用 $\phi: \mathbf{X} \to \mathbf{H}$ 表示从输入空间 $\mathbf{X} \subset \mathbf{R}^m$ 到特征空间 $\mathbf{H}$ 的一个非线性变换，假设在特征空间中问题是线性可分的，对应最优超平面如下：

$$\mathbf{w}^{\phi T} \phi(\mathbf{x}) + b = 0 \tag{3.22}$$

在不失去一般性，设偏移量 $b=0$，式(3.22)简化为：

$$\mathbf{w}^{\phi T} \phi(\mathbf{x}) = 0 \tag{3.23}$$

与线性可分情况相似，可以用拉格朗日乘子法在特征空间中寻找最优权重向量 $\mathbf{w}^{\phi *}$：

$$\mathbf{w}^{\phi *} = \sum_{i=1}^{n} \alpha_i^* y_i \phi(\mathbf{x}_i) \tag{3.24}$$

在特征空间中的最优超平面如下：

$$\sum_{i=1}^{n} \alpha_i^* y_i \phi^T(\mathbf{x}_i)\, \phi(\mathbf{x}) = 0 \tag{3.25}$$

$\phi^T(\mathbf{x}_i)\phi(\mathbf{x})$表示了$\phi^T(\mathbf{x}_i)$和$\phi(\mathbf{x})$的内积，由此可导出内积核函数：

定义 3.3.1(内积核)[7]　核是一个函数 $K(\mathbf{x}, \mathbf{x}')$，对所有的 $\mathbf{x}, \mathbf{x}' \in \mathbf{X} \subset \mathbf{R}^m$，满足：

$$K(\mathbf{x}, \mathbf{x}') = \phi^T(\mathbf{x})\, \phi(\mathbf{x}') \tag{3.26}$$

其中ϕ为从输入空间 $\mathbf{X}$ 到特征空间 $\mathbf{H}$ 的变换。

核的重要性在于：我们可以用它在特征空间中构建最优超平面，而不用考虑变换ϕ的具体形式，通常也不需要在高维(甚至无限维)特征空间中对其进行显示形式化。因此，使用核算法对维数不敏感，为在高维的空间训练一个线性分类器以有效解决原本线性不可分的问题创造了条件。在式(3.25)中用 $K(\mathbf{x}_i, \mathbf{x})$ 替代$\phi^T(\mathbf{x}_i)\phi(\mathbf{x})$，即可得到如下最优超平面：

$$\sum_{i=1}^{n} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}) = 0 \tag{3.27}$$

核技巧引人入胜之处是它可以简化计算，避免直接在复杂的特征空间上计算内积计算和设计分类器。

在运用核技巧之前先要考虑如何构建核函数，或说核函数应该满足何种特性。为了解答这个问题，我们先引入 Mercer 定理，它描述了一个核函数 $K(\mathbf{x}, \mathbf{x}')$应该具备的性质。

定理 3.3.2Mercer 定理[13] 令 $\mathbf{K}(\mathbf{x},\mathbf{x}')$是一个在闭区间 $a\leqslant\mathbf{x}\leqslant b$($\mathbf{x}'$也一样)上的连续对称核函数,该核函数 $\mathbf{K}(\mathbf{x},\mathbf{x}')$可以展开为以下级数:

$$\mathbf{K}(\mathbf{x},\mathbf{x}')=\sum_{i=1}^{\infty}\lambda_i\varphi_i(\mathbf{x})\varphi_i(\mathbf{x}') \tag{3.28}$$

所有系数为正 $\lambda_i>0$。该级数是有效和收敛的充要条件是:

$$\int_b^a\int_b^a\mathbf{K}(\mathbf{x},\mathbf{x}')\psi(\mathbf{x})\psi(\mathbf{x}')d\mathbf{x}d\mathbf{x}'\geqslant 0 \tag{3.29}$$

所有的 $\psi(\cdot)$都满足条件:

$$\int_b^a\psi^2(\mathbf{x})d\mathbf{x}<\infty \tag{3.30}$$

从这一定理可总结出对核函数(称为 Mercer 核)最重要的特征是,对于输入空间 $\mathbf{x}$ 的任意随机有限子集,由核函数 $\mathbf{K}(\mathbf{x},\mathbf{x}')$构造的矩阵(称为 Gram 矩阵)

$$\mathbf{K}=(\mathbf{K}(\mathbf{x}_i,\mathbf{x}'_j))_{i,j=1}^n \tag{3.31}$$

是一个半定对称矩阵[7]。

根据这一要求,在实践中核的选择有一定的自由度。例如,除了线性核函数,我们还可以定义多项式或径向基核函数。近年来,人们对可用于 SVC 分类以及其他统计测试问题的不同核函数进行了很多研究,我们在后面的部分会讨论这些核。

在 3.2 节我们介绍了用软间隔 SVC 求解线性不可分问题,显然这同核技巧解决线性不可分问题的方式不同。软间隔是放松对输入空间中的限制,允许某些错误存在。但是,当问题极度线性不可分以及分类错误发生次数过高的时候,软间隔方法就行不通了。核技巧则是通过核函数将数据隐式地映射到高维的特征空间,以使得线性不可分变成可分。当然由于实际问题的复杂性,核技巧并不总能保证问题能线性可分。所以,在实践中我们通常会把这两种方法结合在一起以发挥它们各自的优势,更有效地解决线性不可分问题。核软间隔 SVC 的受限最优化问题的对偶形式如下[7, 8, 9]:

$$\begin{cases}\max\limits_{\alpha}W(\alpha)=\sum\limits_{i=1}^{n}\alpha_i-\dfrac{1}{2}\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}\alpha_i\alpha_j y_i y_j\mathbf{K}(\mathbf{x}_i,\mathbf{x}_j)\\ \sum\limits_{i=1}^{n}\alpha_i y_i=0\quad 0\leqslant\alpha_i\leqslant C,\quad (i=1,\cdots,n)\end{cases} \tag{3.32}$$

采用拉格朗日乘子法,我们可以得到最优分类器:

$$f(\mathbf{x})=\sum_{i=1}^{n}\alpha_i^*\,y_i\mathbf{K}(\mathbf{x}_i,\mathbf{x})+b^* \tag{3.33}$$

其中 $b^*=1-\sum\limits_{i=1}^{n}\alpha_i^*\,y_i\mathbf{K}(\mathbf{x}_i,\mathbf{x}_s)$,对正的支持向量有 $y_s=+1$。

例 3.3.3 (说明性示例)异或问题(XOR)是一个典型的线性不可分问题。我们用它来说明核技巧和软间隔 SVC 结合在复杂分类问题中的重要性。本例中,用四个不同的高斯分布随机生成一个二维 XOR 数据集,用"*"和"·"表示两个类别的样本。

如图 3.2(a)所示,在 XOR 问题上基于线性核的硬间隔 SVC 完全无效。线性边界虽然将所有的样本分成两部分,但是这种划分并不能将两个类别分开,没有实现分类的目标。接下来,我们使用软间隔 SVC 与径向基核相结合来求解这个问题:

$$K(x_i, x) = \exp\left(-\frac{||x - x_i||^2}{\sigma^2}\right)$$

我们将正则化参数设置为 C=1，核参数或带宽设置为 $\sigma=1$，得到的判别边界如图 3.2(b)所示。通过使用核技巧，曲线形的分类边界将两个类别中一个类别的数据包围了起来。现在分类器只需判断样本处于边界的内部还是外部就可以对样本进行正确的分类。

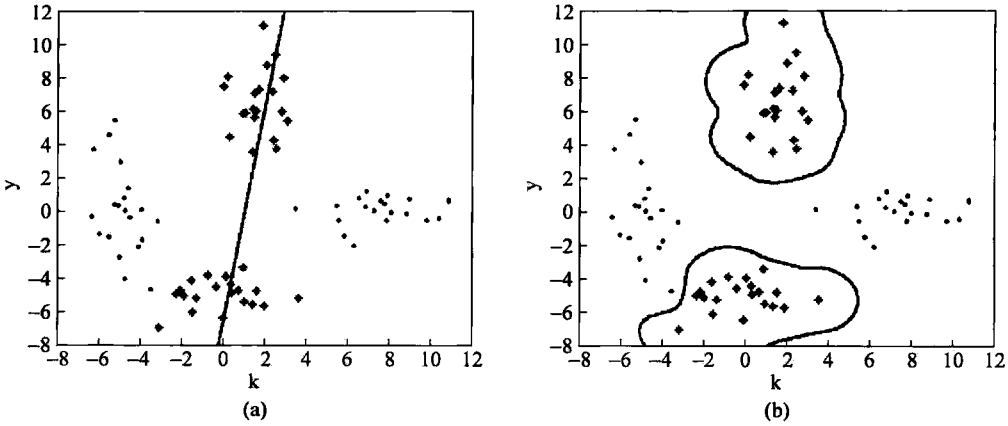


图 3.2　SVC 在 XOR 问题的判别边界

(a)使用线性核的硬间隔 SVC；(b)使用径向基核的软间隔 SVC

例 3.3.4 实际应用示例　SVC 算法已广泛应用于许多重要的科学领域，如生物信息学、物理学、化学、医学和天文学等。本例中，我们从 UCI 机器学习仓库(http://www.ida.first.fraunhofer.de/projects/bench/benchmarks.htm)精心选择了五个医疗领域的数据集来说明 SVC 的实际应用。

这 5 个数据集分别是 B.-cancer(威斯康星州的乳腺癌数据)，Diabetes(Pima 印第安人糖尿病数据)，Heart(心脏数据)，Thyroid (甲状腺疾病数据)和 Splice (拼接交叉基因序列数据)。

表 3.1 的第 2～4 列给出了数据集的信息，分别是样本的维数、训练集的样本数量和测试集的样本数量；第 5～8 列显示的是实验信息，分别是最优正则化参数 C、最优核参数 σ、支持向量的平均数量 SV 和带方差的分类精度 Accruacy，其中最优正则化参数 C 和最优核参数 σ 使用交叉验证方法确定的。在前 4 个数据集上，独立运行算法 100 次；在第 5 个数据集上，独立运行算法 200 次。

表 3.1　SVC 算法在 5 个数据集上的结果

数据集	维数	训练集	测试集	C	σ	SV	精度
B.-cancer	9	200	77	1.519e+01	5.000e+01	138.80	0.7396±4.74
Diabetes	8	468	300	1.500e+01	2.000e+01	308.60	0.7647±1.73
Heart	13	170	100	3.162e+00	1.200e+02	86.00	0.8405±3.26
Thyroid	5	140	75	1.000e+01	3.000e+00	45.80	0.9520±2.19
Splice	60	1000	2175	1.000e+03	7.000e+01	762.40	0.8912±0.66

从表 3.1 可以看到，支持向量的数量比训练样本的数量明显要少，这显示了 SVC 算法

具有很好的稀疏性;精度值较高则显示了 SVC 算法分类性能很好;方差较低表明在实际应用中 SVC 算法有良好的稳定性。

3.4 理论基础

在前面几节中,我们首先描述了线性可分和线性不可分情形下的 SVC 算法,进一步,为了提高分类器的表示能力引入了核技巧,使得学习算法可以抓住高维特征空间中的线性性质同时又避免了维数灾难。本节我们将讨论 SVC 的理论基础,首先基于 VC 理论[4,5]介绍一个通用的线性分类器错误界,它可以从宏观上指导我们如何去控制分类器的复杂度。接下来我们推导一个 SVC 的具体的泛化界,它可以解释 SVC 的最大间隔性质为什么可以确保分类器良好泛化能力。

VC 理论推广了统计学习中的概率近似正确(PAC)学习模型并直接导致了 SVM 的出现。该理论提出的泛化界可以用来估计泛化错误,关键在于它定义了一种新的复杂性度量——VC 维这一概念。

假设训练数据和测试数据都是由一个确定但未知的概率分布 **D** 生成的,那我们可以定义 **D** 上一个分类函数 h 的错误 $\mathrm{err}_{\mathbf{D}}(h)$为:

$$\mathrm{err}_{\mathbf{D}}(h) = \mathbf{D}\{(\mathbf{x},y):h(\mathbf{x}) \neq y\} \tag{3.34}$$

它度量的是期望错误[7]。

PAC 模型确定了泛化错误随机变量的分布的界 $\mathrm{err}_{\mathbf{D}}(h_s)$,界的形式为 $\varepsilon=\varepsilon(n,H,\delta)$,即假设 h_s 在训练数据 S 上的错误率满足[7]:

$$\mathbf{D}^n\{S:\mathrm{err}_{\mathbf{D}}(h_s) > \varepsilon(n,H,\delta)\} < \delta \tag{3.35}$$

如果有|H|个假设在 S 上有大量错误,那么 PAC 界为:

$$\varepsilon = \varepsilon(n,H,\delta) = \frac{1}{n}\ln\frac{|H|}{\delta} \tag{3.36}$$

定理 3.4.1Vapnik and Chervonenkis[7] 令假设空间 H 的 VC 维是 d,对于 $\mathbf{X}\times\{-1,1\}$上的随机概率分布 **D**,在训练集 S 上随机假设 $h\in H$ 的泛化错误以概率 $1-\delta$ 满足:

$$\mathrm{err}_{\mathbf{D}}(h) \leqslant \varepsilon(n,H,\delta) = \frac{2}{n}\left(\log\frac{2}{\delta} + d\log\frac{2en}{d}\right) \tag{3.37}$$

其中,$d\leqslant n$,$n>2/\varepsilon$。

式(3.37)的第一项是训练错误,第二项同 d 成比例。所以,该定理表明在假设 h 将经验错误控制在较小范围的前提下,如果能最小化 d 就可以最小化预测错误。

定理 3.4.1 为线性分类器提供了一个通用的界,从而给出如何控制分类器复杂度的宏观指导。接下来,我们将这个界在 SVC 上进行运用和改进,最终推出 SVC 的泛化错误的界。

我们首先给出间隔的正式定义。

定义 3.4.2(间隔)[7] 用实值函数类 F 对输入空间 **X** 进行分类(以 0 为分界),实例 $(\mathbf{x}_i,y_i)\in\mathbf{X}\times\{-1,1\}$到函数或超平面 $f\in F$ 的间隔为:

$$\gamma_i = y_i f(\mathbf{x}_i) \tag{3.38}$$

$\gamma_i>0$ 就表示实例$(\mathbf{x}_i,y_i)$被正确地分类了。训练集 S 关于 f 的间隔分布是指 S 中的实例关

于 f 的间隔的分布，其极小值记为 $m_S(f)$。

虽然 VC 维中的 d 有理论意义，但是很多时候 d 是无限的，这导致很多实际问题不能使用该泛化界。因此，我们在 SVC 中引入一种与间隔有关的相似性度量，以取代传统的 VC 维。

定义 3.4.3(函数的覆盖)[7]　设 F 是在 X 上的实值函数类。对输入数据序列：

$$S=\{\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_n\}$$

F 的 γ-覆盖是一个有限函数集 B，对于所有的 $f\in F$，存在一个 $g\in B$，满足 $\max_{1\leqslant i\leqslant n}(|f(\mathbf{x}_i)-g(\mathbf{x}_i)|)<\gamma$。令 $N(F,S,\gamma)$为覆盖的最小尺寸，F 覆盖的数据的数量定义为：

$$N(F,n,\gamma)=\max_{S\in\mathbf{X}^n}N(F,S,\gamma) \tag{3.39}$$

当假设 f 在训练集 S 上满足 $m_s(f)=\gamma$ 时，可以使用 $N(F,n,\gamma)$来重写定理 3.4.1。

定理 3.4.4 带间隔的 VC 定理[7]　考虑一个有界实值函数空间 F 和固定的 $\gamma\in\mathbf{R}^+$。对于在 $\mathbf{X}\times\{-1,1\}$上任意的概率分布 **D**，训练集 S 上边界为 $m_s(f)\geqslant\gamma$ 的假设 $f\in F$ 的泛化错误以 $1-\delta$ 的概率满足：

$$\mathrm{err}_D(f)\leqslant\varepsilon(n,F,\delta,\gamma)=\frac{2}{n}\left(\log\frac{2}{\delta}+\log N\left(F,2n,\frac{\gamma}{2}\right)\right) \tag{3.40}$$

其中 $n>\frac{2}{\varepsilon}$。

定理 3.4.4 展示了如何使用 $m_s(f)$来限定通过训练数据获得的泛化错误，$N\left(F,2n,\frac{\gamma}{2}\right)$可以看作另一种形式的 VC 维，$\gamma$ 越大 $N\left(F,2n,\frac{\gamma}{2}\right)$越小。我们可以得出一个结论：对于小规模样本，大间隔可以确保分类器有好的泛化性能。

虽然定理 3.4.4 已经推广了定理 3.4.1，但在实际应用中 $N\left(F,2n,\frac{\gamma}{2}\right)$的具体值仍然很难确定。因此，我们将进一步为特定 SVC 算法推导出一个更具体的错误界。

定理 3.4.5　SVC 的泛化界[7]　假设输入空间 **X** 是内积空间 **H** 中的一个半径为 R 的超球，即 $\mathbf{X}=\{\mathbf{x}\in\mathbf{H}:||\mathbf{x}||_{\mathbf{H}}\leqslant R\}$。

考虑函数类 ψ：

$$\psi=\{\mathbf{x}\mapsto\mathbf{W}^{\mathrm{T}}\mathbf{X}:||\mathbf{w}||_{\mathbf{H}}\leqslant 1,\mathbf{x}\in\mathbf{X}\}$$

$\gamma\in\mathbf{R}^+$是固定的。对于 $\mathbf{X}\times\{-1,1\}$上的概率分布 **D**，训练集 S 上的间隔 $m_s(f)\geqslant\gamma$ 的假设 $f\in\psi$ 的泛化错误以 $1-\delta$ 的概率满足：

$$\begin{aligned}\mathrm{err}_D(f)&\leqslant\varepsilon(n,\psi,\delta,\gamma)\\&=\frac{2}{n}\left(\log\frac{4}{\delta}+(64R^2/\gamma^2)\left(\log\frac{en\gamma}{4R}\right)\left(\log\frac{128nR^2}{\gamma^2}\right)\right)\end{aligned} \tag{3.41}$$

其中 $n>\frac{2}{\varepsilon}$，$64R^2/\gamma^2<n$。

特别要注意的是输入空间的维数没有出现在这个界里，因此，这个界可以用于任意维数的空间，也就意味着这个界可以克服维数灾难。此外，当样本分布良好时，这个界以很高的概率确保随机测试样本预测错误很少。在这种情况下，间隔 γ 可以看成是样本分布质量的一个度量，因此可以进一步的度量 SVC 算法的泛化性能[7]。

3.5 支持向量回归器

到目前为止，我们关注的是分类任务的 SVC 方法。在本节中，我们将考虑使用 SVM 去解决非线性回归问题 SVR。与分类算法相似，将线性学习方法和核技巧结合即可使用非线性函数来挖掘和利用最大间隔方法优势，并且确保相关算法在高维情况下仍然是有效的[7]。

对于回归问题，如果数据存在离群点或者基础分布有长尾型的噪声，传统的最小二乘法估计的性能就会严重下降[13]。因此，我们需要一种健壮的估计量，对于模型微小变化不敏感，这就是我们所说的 ε—不敏感损失函数。

定义 3.5.1(ε—不敏感损失函数) 令 f 是 $\mathbf{X}$ 上的实值函数，ε—不敏感损失函数 $L^{\varepsilon}(\mathbf{x},y,f)$ 定义如下：

$$L^{\varepsilon}(\mathbf{x},y,f) = |\ y-f(\mathbf{x})\ |_{\varepsilon} = \max(0,\ |\ y-f(\mathbf{x})\ |-\varepsilon) \tag{3.42}$$

如果输出 $f(\mathbf{x})$ 的估计与期望响应 y 的差值的绝对值小于 ε 或者等于 0，那么 $L^{\varepsilon}(\mathbf{x},y,f)=0$，否则等于 $f(\mathbf{x})$ 的估计与期望响应 y 的差值减去 ε 再取绝对值。

现在考虑一个非线性回归模型：

$$y = g(\mathbf{x}) + v \tag{3.43}$$

加性噪音 v 与输入向量 $\mathbf{x}$ 是统计独立的，函数 $g(\cdot)$ 和噪音 v 的统计量是未知的，知道的是训练数据集

$$S=\{(\mathbf{x}_1,y_1),\cdots,(\mathbf{x}_n,\ y_n)\}$$

和一个函数类

$$F=\{f(\mathbf{x})=\mathbf{w}^{\mathrm{T}}\mathbf{x}+b,\mathbf{w}\in\mathbf{R}^m,b\in\mathbf{R}\}$$

此处的目标是确定适当的参数值 $\mathbf{w}$ 和 b，从而使 $f(\mathbf{x})$ 逼近未知目标函数 $g(\mathbf{x})$。原问题可以表示为：

$$\begin{cases}\min\limits_{\mathbf{w},b}\dfrac{1}{2}\|\mathbf{w}\|^2+C\sum\limits_{i=1}^{n}(\xi_i+\hat{\xi}_i)\\ (\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)-y_i\leqslant\varepsilon+\xi_i, & (i=1,\cdots,n)\\ y_i-(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)\leqslant\varepsilon+\hat{\xi}_i, & (i=1,\cdots,n)\\ \xi_i,\hat{\xi}\geqslant 0 & (i=1,\cdots,n)\end{cases} \tag{3.44}$$

使用拉格朗日乘子法，得到对偶问题是：

$$\begin{cases}\max\limits_{\alpha,\hat{\alpha}}W(\alpha,\hat{\alpha}) = \sum\limits_{i=1}^{n}y_i(\hat{\alpha}_i-\alpha_i)-\varepsilon\sum\limits_{i=1}^{n}(\hat{\alpha}_i+\alpha_i)-\dfrac{1}{2}\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}(\hat{\alpha}_i-\alpha_i)(\hat{\alpha}_j-\alpha_j)\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j\\ \sum\limits_{i=1}^{n}(\hat{\alpha}_i-\alpha_i)=0 \quad 0\leqslant\alpha_i,\hat{\alpha}_i\leqslant C, \quad (i=1,\cdots,n)\end{cases} \tag{3.45}$$

我们还可以再进一步在式(3.45)中引入内积核，进而将回归算法扩展到特征空间中。在核空间(即特征空间)中的线性学习器可以获得具有非线性能力的函数。

SVR 比 SVC 多一个自由参数 ε。设置偏移量 $b=0$，自由参数 ε 和 C 就控制了以下逼近函数的 VC 维：

$$f(x) = w^T x = \sum_{i=1}^{n} (\hat{\alpha}_i - \alpha_i) K(x_i, x) \tag{3.46}$$

ε 和 C 应由用户指定，用以直接对回归复杂度进行控制，但如何通过选择 ε 和 C 来获得一个较好的近似函数仍是一个开放问题。

3.6　软件实现

LibSVM[6]和 SVMlight[17]是 SVM 算法的两个最有名的实现。

LibSVM 不仅提供了 Windows 系统下的可执行程序，而且还提供 C++ 和 Java 的源代码以便扩充、改进和移植到其他操作系统。特别是相对于其他 SVM 算法来讲，LibSVM 的可调参数相对较少，而且为了高效处理实际应用提供了大量默认参数。

SVMlight 是用 C 语言实现的。它采用了一种有效集合选择技术，该技术主要是基于最陡可行下降法和另两个有效的计算策略：内核求值的“缩减”和“缓存”。SVMlight 主要包括两个 C 程序：SVM learn——基于训练样本学习分类器；SVM classify——用于分类测试样本。该软件还提供了两种估计泛化性能的评估方法：XiAlpha-估计，基本上没有计算费用但有偏差；Leave-one-out 测试，几乎没有偏差。

除了以上两个实现外，还有大量的完整机器学习工具箱包含了 SVM 算法实现，例如 Torch(C++)、Spider(MATLAB)和 Weka(Java)等，它们都可以从网页 http://www.kernel-machines.org. 上找到。

3.7　当前和未来的研究

过去的十多年里 SVM 在理论和实践上都取得了快速的发展，但未来仍有许多工作要做。在本节，我们考察几个已经取得重大进展的研究方向及这些方向上的开放问题。

3.7.1　计算效率

早期 SVM 的一个缺点是训练阶段的计算复杂性太高，导致算法不适用于大规模数据集，但现在这个问题已经被成功地解决了。一种方法是将大的优化问题分解成一系列小的问题，而每一个小问题又只涉及少数几个精心挑选的变量，每个小问题可以被优化算法高效处理，再通过一个迭代过程对分解得到的子优化问题逐一求解。

一个更近期的方法是将 SVM 的学习问题看成是寻找某个实例集合的近似最小外接球[18～21]。这些实例被映射到 N 维空间后形成一个核心集，在此基础上可以构建一个近似的最小外接球。利用这些核心集上的 SVM 学习问题的解可以很快生成原数据上好的近似解。例如，核心向量机[18]和大球向量机[21]只需几秒钟就可以从数百万的数据中学习出 SVM。

3.7.2　核的选择

对于核 SVM，核函数一般要求满足 Mercer 条件。常用的核函数有三种，分别是 sigmoid 核、多项式核和径向基函数核，但核技巧实际上并不局限于这些核。最近，Pekalska

等提出了一种基于一般近似关系映射[22]的核函数设计的新方法，从而不再要求核函数必须满足 Mercer 条件，也不限定核函数只能有一个特征空间。在实验中这种核表现出比 Mercer 核更好的分类性能，但是与之相关的基础理论需要进一步研究。

此外，另一种流行的方法是多核学习（相对于单一核）。其思想是组合多个核函数来获得更好的结果[23～29]（这类似于集成核）。通过设置恰当的目标函数以及选择合适的核参数，就能实现多个核的混合。

3.7.3 泛化分析

我们一般用 VC 维来估计核机器的泛化错误界，但它含有一个与训练数据无关的固定的复杂度惩罚项，不便推广使用[30]。为了解决这个问题，人们引入了 Rademacher 复杂度代替经典的 VC 维来评估分类器的复杂性[31～34]。其思想比较直观，就是我们可以用分类器拟合随机数据的能力来度量该分类器的能力（或复杂度）。

定义 3.7.1（Rademacher 复杂度） 对于分布 D 在集合 **X** 上生成的样本 $S=\{\mathbf{x}_1,\cdots,\mathbf{x}_n\}$，和定义域是 **X** 的一个实函数类 F，定义 F 的经验 Rademacher 复杂度为一个随机变量：

$$\hat{R}_n(\mathbf{F})=E_\sigma\left[\sup_{f\in F}\left|\frac{2}{n}\sum_{i=1}^{n}\sigma_i f(\mathbf{x}_i)\right|\,\middle|\,\mathbf{x}_1,\cdots,\mathbf{x}_n\right] \tag{3.47}$$

$\sigma=\{\sigma_1,\cdots,\sigma_n\}$ 是独立的取单位值 $\{\pm 1\}$ 的（Rademacher）随机变量。定义 F 的 Rademacher 复杂度为：

$$R_n(\mathbf{F})=E_S[\hat{R}_n(\mathbf{F})]=E_{S\sigma}\left[\sup_{f\in F}\left|\frac{2}{n}\sum_{i=1}^{n}\sigma_i f(\mathbf{x}_i)\right|\right] \tag{3.48}$$

该期望公式中 sup 那部分用来度量是——函数类中的函数和随机标签之间的最好相关性。进而，在核机器上可以得到 Rademacher 复杂度的上界。

定理 3.7.2 复杂度分析[35] 如果 $k:\mathbf{X}\times\mathbf{X}\to\mathbf{R}$ 是一个核，$S=\{\mathbf{x}_1,\cdots,\mathbf{x}_n\}$ 是一个对 **X** 中点的采样，分类器 $\mathbf{F}_B$ 的经验 Rademacher 复杂度满足：

$$\hat{R}_n(\mathbf{F}_B)\leqslant\frac{2B}{n}\sqrt{\sum_{i=1}^{n}k(\mathbf{x}_i,\mathbf{x}_i)}=\frac{2B}{n}\sqrt{tr(\mathbf{K})} \tag{3.49}$$

B 是分类器中权重 w 向量的界。

值得注意的是 Rademacher 复杂度的界仅与核矩阵的迹有关，而核矩阵是由具体的训练数据确定的。所以 Rademacher 复杂度比传统的 VC 维相比，更适用于控制分类器的复杂度和估计分类器的泛化性能。

3.7.4 结构化支持向量机的学习

SVM 算法的最初动机是最大化类间的间隔[36]，因此，SVM（SVC）通常关注的是不同类样本的可分性而不关心类内先验的数据分布信息。著名的“没有免费的午餐”原理[12]表明不存在一个本质上优于其他方法的模式分类方法，甚至可以说不借助于额外的信息就不能超过随机猜测。所以这类问题中分类器的形式都是由先验信息或大量训练样本决定的。事实上，不同的实际问题中的不同的类别数据有着不同的底层数据结构。对于分类问题特别是分类器的泛化能力，调整判别边界以适应这种结构往往是至关重要的。然而，传统的

SVM 没有特别处理结构信息，导出的无偏决策超平面位于支持向量的中线[36,37]，对于现实问题这种分类器可能不是最优的。

最近出现了比传统 SVM 更关注结构信息的算法，它们提供一种分类器设计的新视角，即分类器的设计充分考虑数据分布的结构。这些算法可以分成两类。第一类方法基于流形学习，它假设数据位于输入空间的一个子流形上。这方面最典型的算法是拉普拉斯支持向量机(LapSVM)[38,39]。首先，用每个类的拉普拉斯图构建一个 LapSVM，然后，将拉普拉斯矩阵数据蕴含的流形结构以一个附加项的形式引入到传统的 SVM 框架中。

第二种方法基于聚类算法，通常可以假设数据会包含若干个聚类这样的先验分布信息。聚类假设一般比流形假设更具普遍性，也产生了若干种大间隔机器。新近出现的结构化大间隔机器(SLMM) [37]，首先用聚类算法捕捉不同类别中的结构信息，然后在确定决策超平面时将传统的欧氏距离替换为马氏距离，从而将结构信息引入到约束条件中。最大概率机(MPM)[41]和最大最小间隔机(M4) [36]等大间隔机器都可以被视作 SLMM 的特例。实验表明 SLMM 分裂性能很好，但它的训练过程对应于一个二次锥规划(SOCP)问题，这比传统 SVM 的二次规划(QP)的计算成本要高很多。此外，该算法还不易被推广到大规模或者多类问题。再后来，又发展出一种名为结构化支持向量机(SSVM)[42]的新算法，该算法用的是传统 SVM 的经典框架。这样，训练过程对应的优化问题仍然是二次规划(QP)，所以解的稀疏性和可伸缩性也得以保持。研究还表明，无论是理论上还是实践上，SSVM 都比 SVM 和 SLMM 有更好的泛化性能。

3.8　习　　题

1. 请思考以下的简单的二分类问题：

$$c_1：(1,1)^T \quad (-1,3)^T \quad (2,6)^T$$

$$c_2：(-1,-2)^T \quad (1,-3)^T \quad (-5,-7)^T$$

(1) 计算其最优超平面和几何间隔。

(2) 找出支持向量。

(3) 使用 Lagrange 成子法在对偶空间中计算最优解。

2. 请思考另一个二分类问题：

$$c_1：(1,1)^T \quad (3,7)^T \quad (5,9)^T$$

$$c_2：(-1,-2)^T \quad (1,6)^T \quad (2,-1)^T$$

用软间隔 SVC 构建最优超平面并在对偶空间中计算对应解。

3. 构建一个与示例 3.3.3 类似的异或(XOR)问题，讨论径向基核的核参数的选择如何影响分类性能。

4. 令 $\mathbf{K}_1$ 和 $\mathbf{K}_2$ 是 $\mathbf{X}\times\mathbf{X}$ 上的核($\mathbf{X}\subseteq\mathbf{R}^n$)，$a\in\mathbf{R}^+$，$f(\cdot)$是 $\mathbf{X}$ 上的实值函数：

$$\phi：\mathbf{X}\to\mathbf{R}^m$$

$\mathbf{K}_3$ 是 $\mathbf{R}^m\times\mathbf{R}^m$ 上的核，$\mathbf{B}$ 是 $n\times n$ 对称半定矩阵。试证以下函数是核函数：

(1) $\mathbf{K}(\mathbf{x},\mathbf{z})=\mathbf{K}_1(\mathbf{x},\mathbf{z})+\mathbf{K}_2(\mathbf{x},\mathbf{z})$

(2) $\mathbf{K}(\mathbf{x},\mathbf{z})=a\mathbf{K}_1(\mathbf{x},\mathbf{z})$

(3) $\mathbf{K}_1(\mathbf{x}, \mathbf{z})=\mathbf{K}_2(\mathbf{x},\mathbf{z})$

(4) $\mathbf{K}(\mathbf{x},\mathbf{z})=f(\mathbf{x})f(\mathbf{z})$

(5) $\mathbf{K}(\mathbf{x},\mathbf{z})=\mathbf{K}_3(\phi(\mathbf{x}), \varphi_{(z)})$

(6) $\mathbf{K}(\mathbf{x}, \mathbf{z})=\mathbf{x}^T\mathbf{B}\mathbf{z}$

5. 请用 VC 维理论论述 SVR 的泛化界。

6. SVC 可直接用于二分类问题，请讨论一下如何将其扩展到多分类问题？

7. 请讨论一下 SVM 算法的稳健性。

8. 请讨论一下 SVC 没有充分使用先验的类内数据分布信息的情形，此时，判别超平面位于支持向量正中间。

参 考 文 献

[1] V. Vapnik. *The Nature of Statistical Learning Theory*, Springer Verlag, 1995.

[2] V. Vapnik. *Statistical Learning Theory*, Wiley, 1998.

[3] B. Schölkopf, C. J. C. Burges, and A. J. Smola. *Advances in Kernel Methods-Support Vector Learning*, MIT Press, 1999.

[4] O. Chapelle, P. Haffner, and V. Vapnik. Support vector machines for histogram based image classification. *IEEE Trans. on Neural Networks*, vol. 10(3.5),1055 1064, 1999.

[5] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, vol. 20,273 297, 1995.

[6] N. Cristianini, C. Campbell, and J. Shawe-Taylor. A multiplicative updating algorithm for training support vector machine. In *Proceedings of the 6th European Symposium on Artificial Neural Networks* (ESANN), 1999.

[7] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and OtherKernel-based Learning Methods*, Cambridge University Press, 2000.

[8] M. S. Kearns, S. A. Solla, and D. A. Cohn. *Advances in Neural Information Processing Systems*, MIT Press, 1999.

[9] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smythand, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*, MIT Press, 1996.

[10] A. J. Smola, P. Bartlett, B. Schölkopf, and C. Schuurmans. *Advances in Large Margin Classifiers*, MIT Press, 1999.

[11] B. Schölkopf. *Support Vector Learning*, R. Oldenbourg Verlag, 1997.

[12] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*, Wiley, 2001.

[13] S. Haykin. *Neural Networks: A Comprehensive Foundation*, Tsinghua University Press, 2001.

[14] V. Cherkassky, X. Shao, F. Mulier, and V. Vapnik. Model complexity control for regression using VC generalization bounds. *IEEE Transactions on Neural Networks*, vol. 10, 1075 1089, 1999.

[15] V. Cherkassky and F. Mulier. Learning From Data: Concepts, *Theory and Methods*, Wiley, 1998.

[16] C.-C. Chang and C.-J. Lin. LibSVM: A library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, 2001.

[17] T. Joachims. Making Large-scale SVM learning practical. *Advances in Kernel Methods-Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola(eds.), MIT Press, 1999.

[18] I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, vol. 6,363 392, 2005.

[19] I. W. Tsang, J. T. Kwok, and K. T. Lai. Core vector regression for very large regression problems. *ICML*, 913 920, 2005.

[20] I. W. Tsang and J. T. Kwok. Large-scale sparsified manifold regularization. NIPS, Vancouver, Canada, 2006.

[21] I. W. Tsang, A. Kocsor, and J. T. Kwok. Simpler core vector machines with enclosing balls. ICML, 2007.

[22] E. Pekalska, P. Paclik, and R. P. W. Duin. A generalized kernel approachto dissimilarity-based classification. *Journal of Machine Learning Research*, vol. 2, 175 211, 2001.

[23] J. Bi, T. Zhang, and K. Bennett. Column-generation boosting methods for mixture of kernels. KDD, 521 526, 2004.

[24] I. M. de Diego, J. M. Moguerza, and A. Munoz. Combining kernel information for support vector classification. Multiple Classifier Systems, 102 111, 2004.

[25] Y. Grandvalet and S. Canu. Adaptive scaling for feature selection in SVMs. *Neural Information Processing Systems*, 2002.

[26] G. R. G. Lanckriet, T. D. Bie, N. Cristianini, M. I. Jordan, and W. S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, vol. 20(3.16),2626 2635, 2004.

[27] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *JMLR*, vol. 5, 27 72, 2004.

[28] C. S. Ong, A. J. Smola, and R. C. Williamson. Learning the kernel with hyperkernels. *JMLR*, vol. 6, 1043 1071, 2005.

[29] Z. Wang, S. Chen, and T. Sun. MultiK-MHKS: Anovel multiple kernel learning algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30(3.2), 348 353, 2008.

[30] P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, vol. 3,463 482, 2002.

[31] P. L. Bartlett. The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, vol. 44(3.2), 525 536,1998.

[32] V. Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions Information Theory*, vol. 47(3.5), 1902 1914, 2001.

[33] V. Koltchinskii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. Technical Report, Departmentof Mathematics and Statistics, University of New Mexico, 2000a.

[34] V. Koltchinskii and D. Panchenko. Rademacher processes and bounding the risk of function learning. In E. Gine, D. Mason, and J. Wellner (ed.), *High Dimensional Probability II*, 443 459, 2000b.

[35] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[36] K. Huang, H. Yang, I. King, and M. R. Lyu. Learning large margin classifiers locally and globally. *ICML*, 2004.

[37] D. S. Yeung, D. Wang, W. W. Y. Ng, E. C. C. Tsang, and X. Zhao. Structured large margin machines: Sensitive to data distributions. *Machine Learning*, vol. 68,171 200, 2007.

[38] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from examples. Department of Computer Science, University of Chicago, Tech. Rep, TR-2004-06, 2004.

[39] M. Belkin, P. Niyogi, and V. Sindhwani. On manifold regularization. In *Proceedings of International Workshop on Artificial Intelligence and Statistics*, 2005.

[40] P. Rigollet. Generalization error bounds in semi-supervised classification under the cluster assumption. *Journal of Machine Learning Research*, vol. 8, 1369 1392, 2007.

[41] G. R. G. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, and M. I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, vol. 3, 555 582, 2002.

[42] H. Xue, S. Chen, and Q. Yang. Structural support vector machine. *The Fifth International Symposium on Neural Networks*, Part I, LNCS5263, 2008.

第 4 章　Apriori

Hiroshi Motoda，Kouzou Ohara

4.1　引言
4.2　算法描述
　　4.2.1　挖掘频繁模式和关联规则
　　4.2.2　挖掘序列模式
　　4.2.3　讨论
4.3　软件实现
4.4　示例
　　4.4.1　可行示例
　　4.4.2　性能评估
4.5　高级主题
　　4.5.1　改进 Apriori 类型的频繁模式挖掘
　　4.5.2　无候选的频繁模式挖掘
　　4.5.3　增量式方法
　　4.5.4　稠密表示：闭合模式和最大模式
　　4.5.5　量化的关联规则
　　4.5.6　其他的重要性/兴趣度度量方法
　　4.5.7　类别关联规则
　　4.5.8　使用更丰富的形式：序列、树和图
4.6　小结
4.7　习题
参考文献

4.1 引 言

很多模式寻找算法，如决策树、分类规则归纳和数据聚类等被广泛应用于数据挖掘领域，它们实际上是由机器学习社区发展起来的。相比这一主流，频繁模式和关联规则挖掘是少有的例外，它们直接推动了数据挖掘的研究并对这个领域产生了巨大影响。其基本算法简单且易于实现，本章将介绍频繁模式和关联规则挖掘中最基础的算法 Apriori 和 AprioriTid [3, 4]，以及 Apriori 在序列模式挖掘中的拓展 AprioriAll [6, 5]，并用原始文献中的例子进行阐释。此外，还基于 UCI 数据挖掘资料库中的数据集[8]用 Apriori 算法的一个开源实现软件对该算法的效率进行了分析[1]。Apriori 这种非常基本的算法处理的数据形式主要局限于市场交易。所以，人们在提高计算效率、寻找更紧致表示和扩展处理数据的类型等方面做了大量研究工作，我们将在高级主题一节选择部分有代表性的重要工作予以介绍。

4.2 算法描述

4.2.1 挖掘频繁模式和关联规则

最流行的数据挖掘方法之一是从一个事务数据集中发现频繁项集并推出关联规则。该问题描述如下，令 $I=\{i_1, i_2, \cdots, i_m\}$ 表示一个项集，D 表示事务集，其中每一个事务 t 是一个项集，即 $t \subseteq I$。每个事务都有一个唯一标识 TID。如果 $X \subseteq t$，就说事务 t 包括 I 的一个子集 X。关联规则是一种蕴含形式 $X \Rightarrow Y$，其中 $X \subset I$，$Y \subset I$ 且 $X \cap Y = \varnothing$。在事务集 D 中，规则 $X \Rightarrow Y$ 的支持度 $s(0 \leqslant s \leqslant 1)$ 是说包含 $X \cup Y$ 的事务占全体事务的百分比；规则 $X \Rightarrow Y$ 的置信度 $c(0 \leqslant c \leqslant 1)$ 是说在包含项集 X 的事务中也包含项集 Y 的事务所占百分比。对于一个给定的事务集 D，关联规则挖掘任务是产生所有不小于用户给定的最小支持度(minsup)和最小置信度(minconf)的关联规则。

寻找频繁项集(支持度不小于 minsup 的项集)不是一件容易的事，因为计算过程涉及的组合爆炸会导致不可接受的计算复杂度。只要得到了频繁项集，就很容易生成置信度不小于 minconf 的关联规则。由 R. Agrawal 和 R. Srikant 提出的 Apriori 和 AprioriTid 算法，是特别适合大型交易数据集的挖掘算法[3]。

4.2.1.1 Apriori

Apriori 算法的功能是寻找所有支持度不小于 minsup 的项集。项集的支持度是指包含该项集的事务所占所有事务的比例。频繁项集就是指满足给定的最小支持度的项集。Apriori 的关键在于它使用了一种分层的完备搜索算法(深度优先搜索)，该搜索算法用到了项集的反向单调性，即：如果一个项集是非频繁的，那么它的所有超集也是非频繁的，这个性质也被称为向下闭合性。该算法会对数据集进行多次遍历：第一次遍历，对所有单项的支持度进行计数并确定频繁项；在后续的每次遍历中，利用上一次遍历所得频繁项集作为种子项集，产生新的潜在频繁项集——候选项集，并且对候选项集的支持度进行计数，在本次遍历结束时统计满足最小支持度的候选项集，本次遍历对应的频繁项集就算是确定了，这些

频繁项集又成为下一趟遍历的种子；重复此遍历过程，直到再不能发现新的频繁项集。

依例，Apriori 假设事务或项集里的各项已经预先按词典序进行排列。项集里项的数量被称为项集的大小，大小为 k 的项集被称为 k-项集。用 F_k 表示大小为 k 的频繁项集的集合，用 C_k 表示相应候选项集的集合，F_k 和 C_k 都包含一个计数支持度的字段。

算法 4.1 给出了 Apriori 算法。第一次遍历只对每个单项的出现次数计数并确定频繁 1-项集。随后的遍历包含两个阶段，第一阶段调用 apriori-gen 函数从第 k－1 次遍历产生的频繁项集 F_{k-1} 中获得 C_k；第二阶段扫描事务集，用函数 subset 对 C_k 中每个候选项集的支持度进行计数。

函数 apriori-gen 以频繁(k－1)项集的集合 F_{k-1} 作为参数，返回包含所有频繁 k-项集的集合的超集。

首先是链接步骤，需要连接 2 个 F_{k-1}：

insert into C_k

select p. itemset_1，p. itemset_2，…，p. itemset_{k-1}，q. itemset_{k-1}

from F_{k-1} p，F_{k-1} q

where p. itemset_1 ＝q. itemset_1，…，p. itemset_{k-2} ＝q. itemset_{k-2}

p. itemset_{k-1} ＜q. itemset_{k-1}

F_kp 指项集 p 是一个频繁 k-项集，p. itemset_k 指频繁项集 p 的第 k 个项。

然后，是约减步骤，考察每个项集 $c \in C_k$，如果 c 存在一个(k－1)-子项集，该子项集没出现在 F_{k-1} 里，则删除项集 c。

函数 subset 以 C_k 和事务 t 为参数，返回事务 t 中包含的所有候选项集。为了加快计数速度，Apriori 将 C_k 中的候选项集存储在哈希树的叶子中。初始状态时，树(根节点)的深度为 1，(除根节点外)每个节点都是叶节点。当叶节点中的项集数量超过预定的阈值，就将叶子节点转换为一个内部节点。深度为 d 的内部节点指向深度为 d＋1 的若干内部节点，选哪条分支是通过在深度为 d 的项集上用哈希函数计算出来的。因此，每个叶节点包含的项集数不超过某个最大值(准确地说，该最大值只有在创建一个深度为 d＜k 的内部节点时才达到)，为了找到叶节点中的项集，需要从根开始对项集里的每个项逐一哈希。一旦哈希树被构建，函数 subset 就可以从根节点开始寻找事务 t 中包含的所有候选项集。在根节点，对 t 中所有的项都进行哈希，每一个哈希都对应深度为 1 的一个分支。如果到达叶节点后，搜索事务 t 在叶节点中的项集就得到了答案集；如果哈希项 i 达到一个内部节点，就继续循环哈希事务 t 中项 i 之后的项直至达到叶节点。因此那些不能达到叶子节点的项集就不在 t 中。

任何一个频繁项集的子集显然也都满足最小支持度。链接操作相当于用数据集里的每个项对 F_{k-1} 进行扩充，条件 p. itemset_{k-1} ＜q. itemset_{k-1} 的作用防止项集里有重复的项。约减操作就是从 C_k 中将不包含在 F_{k-1} 中的含有(k－1)-子项集的项集删除，所有应该在 F_k 中的项集则不会被删除。因此，$C_k \supseteq F_k$，即 Apriori 是正确的。

算法 4.1　Apriori 算法

```
F1＝{frequent 1-itemsets};
for(k＝2; Fk－1≠∅; k＋＋) do begin
    Ck＝apriori-gen(Fk－1);          //新的候选
```

```
    foreach transaction t∈D do begin
        C_t=subset(C_k,t);                    //识别属于 t 的所有候选
        foreach candidate c∈C_t do
            c.count ++;
        end
    F_k={c∈C_k | c.count≥minsup};
end
Answer=∪_k F_k;
```

剩下的任务是根据频繁项集生成关联规则。这项任务的直接算法如下：枚举每个频繁项集 f 的所有非空子集 a，如果 support(f)除以 support(a)不小于 minconf，即生成一条规则 a⇒(f－a)。对任意 â⊂a，规则 â⇒(f－â)的置信度不可能大于规则 a⇒(f－a)的置信度，这意味着如果规则(f－a)⇒a 成立，那么所有(f－â)⇒â 形式的规则都成立。利用这个对偶性质，算法 4.2 给出了关联规则的生成算法。

算法 4.2 关联规则生成算法

```
H_1=∅                                    //初始化
foreach; frequent k-itemset f_k, k≥2 do begin
A=(k－1)-itemsets a_{k-1} such that a_{k-1}⊂f_k;
foreach a_{k-1}∈ A do begin
    conf=support(f_k)/support(a_{k-1});
    if (conf≥mincon f) then begin
        output the rule a_{k-1}⇒(f_k－a_{k-1})
            with confidence=conf and support=support(f_k);
        add(f_k－a_{k-1}) to H_1;
        end
end
call ap-genrules(f_k,H_1);
end
procedure ap-genrules(f_k: frequent k-itemset, H_m: set of m-item consequents)
if(k>m+1) then begin
H_{m+1}=apriori－gen(H_m);
foreach h_{m+1}∈ H_{m+1} do begin
    conf=support(f_k)/support(f_k－h_{m+1});
    if(conf≥mincon f) then
        output the rule f_k－h_{m+1}⇒h_{m+1}
            with confidence=conf and support = support(f_k);
    else
        delete h_{m+1} from H_{m+1};
    end
call ap-genrules(f_k,H_{m+1});
end
```

Apriori 通过减少候选集大小来获得良好的性能。然而，在频繁项集很多或者最小支持度很低的情况下，算法必须生成数量庞大的候选项集并且需要反复扫描数据库来检查数量庞大的候选项集，代价仍然十分高。

4.2.1.2 AprioriTid

AprioriTid 是 Apriori 的一个变体，它不是力图去减少候选项集的数量，而是在第一次遍历之后计算支持度时不再使用数据集 D 而是使用新的数据集$\overline{C_k}$。$\overline{C_k}$中每个元素都具有<TID,{ID}>的形式，其中，TID 是事务的标识符，每个 ID 是事务 TID 中的一个潜在频繁 k-项集(除了 k=1)的标识符。如果 k=1，$\overline{C_1}$其实就是数据集 D，只是将原来 D 中每个项 i 被以项集{i}的形式取代。$\overline{C_k}$的一个元素对应于一个事务 t 即<t. TID,{c∈C_k | c 在 t 中}>。

直观上，当 k 值很大的时候，$\overline{C_k}$会变得比数据集 D 小，因为一些事务可能不包含任何候选 k-项集，这种情况下$\overline{C_k}$就不包含该事务，或者该事务虽然包含极少候选项集，但$\overline{C_k}$的每个元素可能小于该事务包含的项的数量。算法 4.3 给出了 AprioriTid 算法，这里，c[i]代表 k-项集 c 中第 i 个项。

算法 4.3 AprioriTid 算法

```
F1 = {frequent 1-itemsets};
C̄1 = database D;
for(k=2; F_{k-1} ≠ ∅; k++) do begin
Ck = apriori-gen(F_{k-1});        //新的候选集
C̄k = ∅;
foreach entry t∈C̄_{k-1} do begin
    //确定 Ck 中的候选项集包含在标识符为 t. TID 的事务中
    Ct = {c∈Ck | (c-c[k])∈t. set-of-itemsets ∧ (c-c[k-1])∈t. set-of-itemsets};
    foreach candidate c∈Ct do
        c. count ++;
    if (Ct ≠ ∅) then C̄k += <t. TID, Ct>;
end
Fk = {c∈Ck | c. count≥minsup};
end
Answer = ∪k Fk;
```

每个$\overline{C_k}$存储在一个顺序结构中。C_k 中的每个候选 k-项集 c_k 除了支持度外还有两个附加字段：生成器和扩展。生成器字段存储了两个频繁(k-1)-项集的 ID_s，这两个项集链接后生成了 c_k，扩展字段存储所有扩展 c_k 得到的(k+1)-候选项集的 ID_s。当一个候选项集 c_k 由 f_{k-1}^1与 f_{k-1}^2链接生成，它们的 ID 被保存在 c_k 的生成器字段中，c_k 的 ID_s 则添加到 f_{k-1}^1的扩展字段中。$\overline{C}_{k-1}$中给定事务 t 的项集的集合{ID}字段给出了包含在 t. TID 中的所有 k-1 候选项集的 ID_s。对于每一个这样的候选集 c_{k-1}，扩展字段给出 T_k，即所有由 c_{k-1}扩展来的 k-候选项集的 ID_s 集合。对于每个 T_k 中的 c_k，生成器字段给出生成 c_k 的两个项集的 ID。如果这些项集出现在项集的集合{ID}中，c_k 就出现在事务 t. TID 中，那么 c_k 就被添加在 C_t 中。

AprioriTid 虽然有计算$\overline{C_k}$的额外开销，但是它的优点在于当 k 较大时存储$\overline{C_k}$的空间较小。因此，在早期遍历(k 较小)时 Apriori 具有优势，在后期遍历(k 较大)时则 AprioriTid 比较好。由于 Apriori 和 AprioriTid 使用了相同的候选项集生成过程，因此对相同的项集进行计数，这样就可以依序将两种算法结合使用。AprioriHybrid 在最初的遍历中使用

Apriori，当它预计$\overline{C_k}$适合存储在内存中时则使用 AprioriTid 算法。

4.2.2 挖掘序列模式

Agrawal 和 Srikant 将 Apriori 算法做了扩展，使其可以处理序列模式挖掘问题[6]。在 Apriori 算法中没有序列的概念，目的是寻找哪些项出现在一起，其实质是挖掘事务内部的模式。当我们关注和寻找序列模式时，就是在寻找事务之间的模式。

每个事务由序列标识、事务时间和项集构成，约定具有同一标识的序列不能含有多个具有相同事务时间的事务。序列就是项集的有序列表，也可以当成是字符集合的列表，但不是字符列表这么简单。序列长度是指序列中项集的个数，记长度为 k 的序列为 k-序列。不失一般性，可将项集映射到连续整数集，则一个项集 i 可记为$(i_1 i_2 \cdots i_m)$，其中 i_j 表示一个项。进而，一个序列 s 可记为$<s_1 s_2 \cdots s_n>$。说一个序列$<a_1 a_2 \cdots a_n>$包含于另一个序列$<b_1 b_2 \cdots b_m>$ $(n \leqslant m)$，是指存在 $i_1 < i_2 < \cdots < i_n$ 使得 $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \cdots, a_n \subseteq b_{i_n}$。所有具有同一序列标识的事务依据事务时间排序成一个序列(事务序列)，说一个序列标识支持序列 s，是指 s 包含于该标识对应的事务序列中。一个序列的支持度被定义为支持该序列的序列标识的数量与所有序列标识的数量之比。同样，一个项集 i 的支持度被定义为在任一事务中含有项集 i 的序列标识的数量与所有序列标识的数量之比，注意，这个定义与在 Apriori 给出的项集的定义不同。项集 i 和 1-序列<i>有相同的支持度。

给定一个事务数据集 D，序列模式挖掘任务就是：从所有满足用户指定的最小支持度的序列中找到那些最大序列，每一个这样的最大序列就代表了一个序列模式。将满足最小支持度的序列称为频繁序列(不一定是最大)，将满足最小支持度的项集称为频繁项集，简记为 fitemset。显然，任何频繁序列都是频繁项集的列表。

序列模式挖掘算法包括五步：(1)排序、(2)频繁项集、(3)转换、(4)序列、(5)最大化。前三个阶段是预处理，最后一个阶段是后处理。

预处理工作：第一步是排序，对数据集 D 中的事务以序列标识为主键以事务时间为次键进行排序。第二步是频繁项集，对 Apriori 算法的支持度计数方法进行修改，再运行算法获得频繁项集，最后将频繁项集映射为连续整数集，这就使得能够在常数时间内完成判断两个频繁项集是否相等的运算。注意到，这个阶段同时也找到了所有的频繁 1-序列。第三步是转换，每个事务被替换为该事务包含的所有频繁项集。如果一个事务中没有任何频繁项集，在变换后的序列中将不再包含该事务，进而如果一个事务序列中没有任何频繁项集，那么就从整个数据集中删除这个序列，但是该序列仍用于计数运算(即数据集的序列总数不变)。转换过程完成后，原来的事务序列被替换成频繁项集的集合的列表，而频繁项集的集合记为$\{f_1, f_2, \cdots, f_n\}$，其中 f_i 表示一个频繁项集。这个转换的目的是提高测试“哪些给定的频繁序列包含于某一个事务序列中?”的效率。转换后的数据集用 D_T 表示。

核心的工作：序列阶段，要枚举出所有的频繁序列。处理这个问题有两大类算法，全部计数(count-all)和部分计数(count-some)，区别在于计数频繁序列的方式。全部计数型的算法要统计所有的频繁序列，这包括后来必须被抛弃的那些非最大序列；而部分计数型算法考虑到最终目标只是为了获得最大序列，所以对包含于更长序列中的序列不再进行计数。Agrawal 和 Srikant 发明了一种全部计数型算法 AprioriAll 和两种部分计数型算法 AprioriSome 和 DynamicSome。由于篇幅限制，本文只介绍 AprioriAll 算法。

后处理工作：最大化阶段，要从所有频繁序列中提取出最大序列。这里主要是利用哈希树(类似于 Apriori 算法中的 subset 函数)快速找到给定序列中的所有子序列。

4.2.2.1　AprioriAll

该算法的概要在算法 4.4 给出。每一轮都利用上一轮得到的频繁序列来生成候选序列，然后通过对数据集进行一轮遍历计算出这些序列的支持度。在最后一轮中，用候选序列的支持度来决定频繁序列。

算法 4.4　AprioriAll 算法

```
F1 = {frequent 1-sequences};                    // 频繁项集阶段的结果
for (k=2; Fk-1 ≠ ∅; k++) do begin
Ck = apriori-gen-2(Fk-1);                       //新的候选序列
foreach transaction sequence t∈DT do begin
    Ct = subseq(Ck, t);                         //t 包含的候选序列
    foreach candidate c∈Ct do
        c. count ++;
    end
    Fk = {c∈Ck | c. count≥minsup};
end
    Answer = maximal sequences in ∪k Fk;
```

算法 4.4 中的 Apriori-gen-2 函数以频繁(k−1)-序列 F_{k-1} 作为参数，首先，需要执行如下的链接操作：

insert into C_k

select p. $fitemset_1$, p. $fitemset_2$, …, p. $fitemset_{k-1}$, q. $fitemset_{k-1}$

　　from F_{k-1} p, F_{k-1}q

where p. $fitemset_1$ = q. $fitemset_1$, …, p. $fitemset_{k-2}$ = q. $fitemset_{k-2}$

然后，考察每个序列 $c\in C_k$，如果 c 含有不属于 F_{k-1} 的(k−1)-子序列，就将 c 从 C_k 里去掉。算法 4.4 中的 subseq 函数的功能类似于 Apriori 的 subset 函数。候选序列 C_k 存储在哈希树中，这样能快速找到事务序列中所有的候选序列。需要注意的是，转换后的事务序列是一个频繁项集的集合的列表，要求同一集合(即事务)中的所有频繁项集具有同一事务时间，而同一序列标识对应的事务序列中不允许出现具有相同事务时间的事务，这个约束已经暗含在 subseq 函数中了。

4.2.3　讨论

Apriori 算法和 AprioriTid 算法都需要预先指定 minsup 和 minconf，一旦这些值发生改变，就必须重新运行算法，而此前运行得到的任何结果都要被抛弃。如果我们事先确实不能确定合适的阈值，又想在不运行算法的情况下知道结果是如何随着阈值的变化而变化的，那么最好就是生成和计数那些在数据集中至少出现一次的项集，并用一个有效的方法保存它们。需要注意，Apriori 算法会生成原始数据集里不存在的候选项集。

Apriori 算法和 AprioriTid 算法存储候选项集时都用到哈希树，还有另一种常用的数据结构 trie 结构[35, 9]。它的每个深度为 k 的节点对应一个候选 k-项集，该节点存储了第 k

个项和项集的支持度。以 trie 中深度为 k－1 的节点为父节点的两个频繁 k-项集共享了前(k－1)-项集，只要链接这两个兄弟节点就能生成候选项集，再经过约减操作就能在频繁 k-项集下再加深一级。为了找到事务 t 中的候选 k-项集，将事务中每个项依序传递给树，从根开始经过若干分支直到第 k 个项。许多 Apriori 算法的实现不仅用 trie 结构存储候选项集，也用它来存储事务[10，9]。

更进一步，我们甚至可以做到，既不去生成候选项集也不必枚举所有频繁项集。我们将在 4.5 节中讨论这些主题。

Apriori 和几乎所有其他的关联规则挖掘算法都使用两阶段策略：第一阶段挖掘频繁模式，第二阶段生成关联规则，其实这并不是唯一的做法。Webb 的 MagnumOpus 采取另一种策略，可以在挖掘频繁模式的同时立即生成所有关联规则的一个大子集[38]。

还有一些对 Apriori 算法族的直接扩展，比如在算法中引入概念体系和时间约束。广义关联规则(Generalized association rules)生成算法 [30]引入了一个用户指定的概念体系，这样即便是基本概念只能生成非频繁项集，还可以用高层概念来得到频繁项集。基本思想就是将每个事务项的所有祖先也加到事务中，然后再运行 Apriori 算法。再有，为了提高计算效率可以采取多种优化措施，例如包含项 x 及其祖先 $\hat{x}$ 的项集 X 的支持度和项集 $X-\hat{x}$ 的支持度是一样的，这样就不需要再算一遍了。广义序列模式(Generalized Sequential Patterns)挖掘算法 GSP[32]不但用到了概念体系，还引入了时间约束。时间约束一方面限制了序列模式中相邻元素(即项集)间的最短和/或最长的时间段，另一方面放宽了“序列元素中所有项必须来自同一事务”的限制，允许同一序列标识下的多个事务中的项用于一个元素中，只要其事务时间在用户指定时间窗口内。该算法还能发现所有的频繁序列模式(而不限于最大序列模式)。GSP 算法运行速度要比 AprioriAll 快 20 倍左右，其中一个原因是 GSP 比 AprioriAll 计数更少的候选集。

4.3 软件实现

Apriori 算法已被很多免费或商业软件实现。本文将介绍三个著名的实现，它们都可以从网上下载。

第一个要介绍的是由 Waikato 大学开发的著名开源机器学习和数据挖掘工具 Weka [40]，这套工具中包含了 Apriori 算法的实现。利用 Weka 的通用图形用户界面，可将 Apriori 算法与 Weka 中很多其他算法一起结合使用。该实现也包括 Weka 自身的扩展，例如，minsup 从上界 Uminsup 到下界 Lminsup 以区间宽度 δminsup 迭代下降。进一步，除了置信度外，还可以用 lift、leverage 和 conviction 等指标来评估关联规则。lift、leverage 将在 4.5 节中讨论。conviction 将用来度量关联规则在考虑蕴含时的独立程度[11]。在使用这些指标时，必须将其最低值都作为阈值。

第二个要介绍的是 Christian Borgelt 的实现[1]，该实现依据 GNU Lesser (Library) General Public License 进行分发，以命令行形式交互，也有一些独立图形用户界面。它很严格地遵循了原始 Apriori 算法系列，还进行了一些拓展，使得该软件运行更快更省内存。该实现用前缀树(一种 Trie 树)存储事务和项集以实现高效的支持度计数[10]。前缀树与 4.2.3 节提到的 trie 树差别不大，用户也可简单地用列表来代替前缀树存储事务。此外，该

实现不仅寻找频繁项集和关联规则，还能寻找闭合项集和最大项集。4.5 节将会讨论闭合项集和最大项集。此外，该实现中除了置信度之外，还可使用信息增益等指标来评估和选择关联规则。

第三个要介绍的是 Frence Bodon 的实现[2]，这是一个用于研究的免费软件。该实现也是基于 Trie 树的，但是 Trie 树的数据结构更为简单，而且只处理频繁项集和关联规则。这是一种命令行程序，输入有四个参数：前三个参数分别是一个包含事务的输入文件、输出文件和 minsup 值；第四个参数 minconf 是可选的，该参数如果为空，程序就只输出频繁项集，如果设定该参数，那么还要执行关联规则的挖掘。这个实现使用面向对象语言 C++ 编写，非常便于快速建立基于 Apriori 算法的应用程序。

4.4 示 例

4.4.1 可行示例

本节我们将用表 4.1 所示的小数据集来讲解前面提到的算法的细节行为。表中 SID 和 TT 列分别表示序列 ID 和事务时间。在讲解关联规则(包括频繁项集)挖掘和最大序列模式挖掘时都是用该数据集，只不过在关于关联规则(包括频繁项集)挖掘的例子中不考虑 SID 和 TT 这两列。

表 4.1 本例用到的事务数据集

TID	SID	TT	项 集
001	1	May 03	c,d
002	1	May 05	f
003	4	May 05	a,c
004	3	May 05	c,d,f
005	2	May 05	b,c,f
006	3	May 06	d,f,g
007	4	May 06	a
008	4	May 07	a,c,d
009	3	May 08	c,d,f,g
010	1	May 08	d,e
011	2	May 08	b,d
012	3	May 09	d,g
013	1	May 09	e,f
014	3	May 10	c,d,f

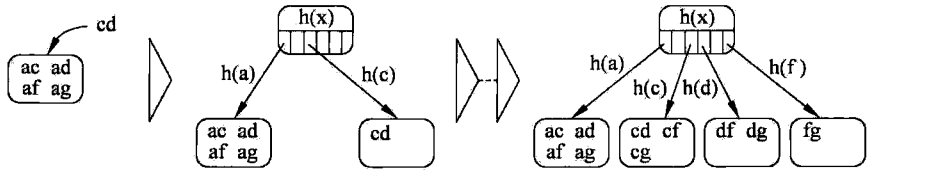
4.4.1.1 频繁项集和关联规则挖掘

本例中，我们要在 minsup=0.6 时找出频繁项集，并在 minconf=0.6 时找出关联规则。

Apriori(算法 4.1)

首先，Apriori 算法扫描整个数据集并导出全部频繁 1-项集 $F_1=\{a,c,d,f,g\}$，至少要 3 个事务才能包含 1-项集的事务。在 F_1 的基础上，调用 apriori-gen 获得候选频繁 2-项集 $C_2=\{ac,ad,af,ag,cd,cf,cg,df,dg,gf\}$，$C_2$ 包含了 F_1 中元素的所有可能的配对，这个阶段没有约减操作。

接着，Apriori 算法对数据集进行扫描并调用 subset 函数计算支持度，该函数用到了哈希树，图 4.1 简要地描述了哈希树的构建和使用的过程。假设将 C_2 中的元素按照字典序添加到哈希树中，每个叶节点可以存放的项集数量的最大值设为 4。当第 5 个项集 cd 被添加到根节点(也是叶节点)中时就会超过阈值。这时就将该节点转变为内部节点，然后再用一个哈希函数 h(x)将每个项集根据哈希值分配到对应的新叶节点中，此处哈希函数的参数 x 是一个项，也就是每个项集的第一个项。我们假设 h(x)是预先确定的且对所有节点都一样。前 4 个项集共享同一首项 a，所以它们被存储到同一个节点中了，而项集 cd 的时候则被分到另外一个叶节点中去了。当我们检查哪些候选被包含在某一个事务(比如事务 004)中时，在根节点处对该事务的每个项进行哈希，如图 4.1(b)所示。首先，对 cdf 中的项 c 哈希到达左边第二个叶节点，就可发现项 cd 和 cf 是 cdf 的子集；接着，对项 d 哈希，发现 df 在左边第三个叶节点(中间的树)中；但哈希项 f 时，在最右边的叶节点里没有发现 cdf 的子集(右边的树)。因此，项集 cd、cf、df 的支持计数值都被加 1。注意，当所有的事务都被处理过后，发现候选 af 和 ag 的出现频次都是 0，这意味着，Apriori 算法可能会生成不存在于原始数据集里的候选项集。



(a) 构建哈希树

(b) 保证集合里面每个元素都有对应的存储位置

图 4.1 哈希树例子

经过对支持度进行计数我们得到了 $F_2=\{cd, cf, df, dg\}$。这些频繁 2-项集将作为推导频繁 3-项集的种子。项集 cd 和 cf 在 F_2 中共享第一个字母 c，二者被 apriori-gen 函数链接生成一个新的候选项集 cdf，因为 df 也存在于 F_2 中。项集 df 和 dg 链接生成的新候选项

又被抛弃，因为其子集 fg 不包括在 F_2 中。因此，候选频繁 3-项集的集合 C_3 仅包含一个项集 cdf。然后，Apriori 算法再次扫描数据集并计算该项集的支持度，就得到了频繁项集 $F_3=\{cdf\}$。由于 F_3 中只有一个项集，所以不能再生成候选频繁集 4-项集，算法也就终止了。

AprioriTid(算法 4.3)

从上面我们看到，Apriori 算法为了获得频繁项集共扫描了数据集 3 次，下面我们将看到 AprioriTid 算法只需扫描数据集 1 次。该算法对 C_2 和 C_3 中的候选项集的支持度进行计算时用到新数据集 $\overline{C}_1$ 和 $\overline{C}_2$。图 4.2 将简要描述了 AprioriTid 算法是如何从这些数据集中找到频繁项集的。$\overline{C}_2$ 是通过计算 C_2 中每个候选项集的支持度得到的，而 $\overline{C}_1$ 是直接通过数据集得到的。假设 $t=<001,\{\{c\},\{d\}\}>\in\overline{C}_1$，接下来 C_2 中的候选项集 cd 被添加到集合 C_t 中，因为 t 的项集的集合$\{\{c\},\{d\}\}$中包含了构成项集 cd 的两个 1-项集。更确切地说，cd 之所以被添加到 C_t 中是因为它是 t 里两个 1-项集的联合，也就是说事务 001 支持 cd，除此以外 C_t 中再没有别的候选项集了，因为事务 001 不能支持 C_2 中其他候选项集了。所以 cd 的支持度就被加 1，$<001,\{\{cd\}\}>$也被加到 $\overline{C}_2$ 中。同理，因为事务 003 支持 $ac\in C_2$，所以$<003,\{\{ac\}\}>$被添加到 $\overline{C}_2$ 中。由于事务 002 不支持任何 2-项集，所以 $\overline{C}_1$ 中的$<002,\{\{f\}\}>$不会被添加到 $\overline{C}_2$ 中。最后，如图 4.2 所示 $\overline{C}_2$ 总共有 9 个条目，比原始数据集小。用同样的方法对候选集 C_3 进行支持度计算可以得到 $\overline{C}_3$，C_3 中有唯一的项集 cdf，$\overline{C}_2$ 中仅有 3 个条目保留到了 $\overline{C}_3$ 中。注意，因为 C_4 是空的，所以不会再使用 $\overline{C}_3$ 了。

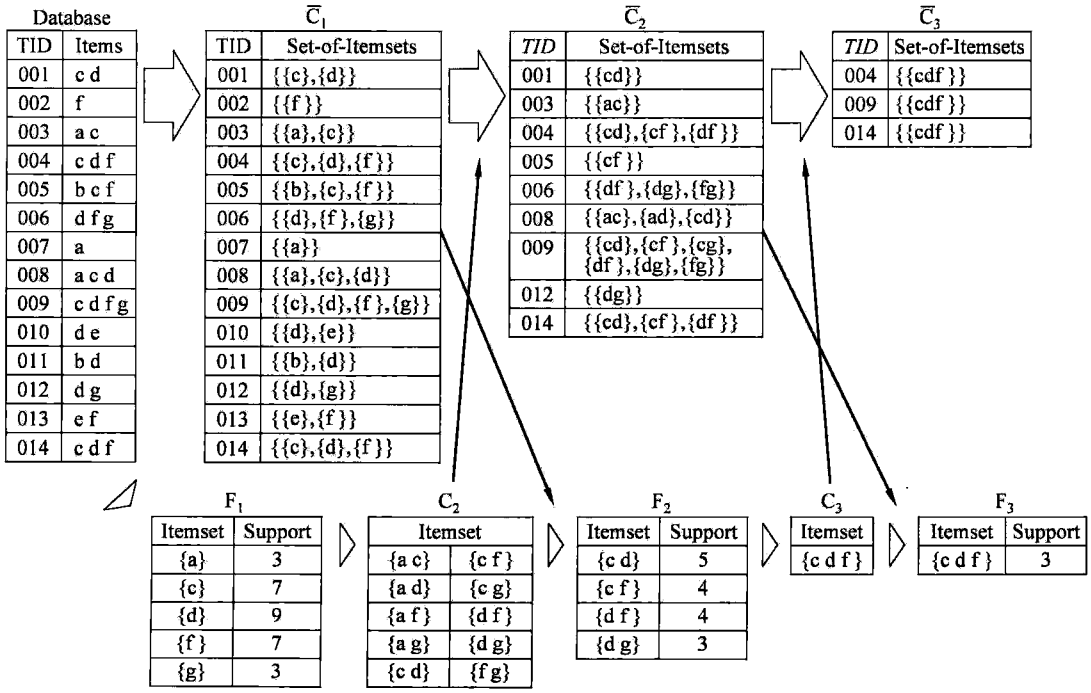


图 4.2　AprioriTid 例子

关联规则(算法 4.2)

本节我们用算法 4.2 从前面得到的频繁项集出发来生成关联规则，此处设定算法参数

mincof=0.6。

我们先来看频繁 2-项集 cd、cf、df、dg，每个频繁项集只能产生两种规则，表 4.2 汇总了这些规则及其置信度。算法 4.2 的输出的关联规则是 1 和 8，因为它们满足 minconf 约束。对每条满足约束的规则都调用一次 ap-genrules 过程，但是因为它不再从频繁 2-项集里生成其他规则，所以也就没有输出。

表 4.2 从频繁 2-项集产生关联规则

序号	规则	置信度	序号	规则	置信度
1	c⇒d	0.71	5	d⇒f	0.44
2	d⇒c	0.56	6	f⇒d	0.57
3	c⇒f	0.57	7	d⇒g	0.33
4	f⇒c	0.57	8	g⇒d	1.0

我们接下来看算法 4.2 从频繁 3-项集 cdf 生成关联规则。首先，如表 4.3 的左半部分所示生成 3 个满足 minconf 约束的关联规则，这些规则的后件的形式是 1-项。然后，在过程 ap-genrules 中用参数 cdf 和{c,d,f}调用 apriori-gen 过程可得到一些 2-项集{cd,cf,df}，这些 2-项集被用作新关联规则的后件，如表 4.3 的右半部分所示。但这几条规则的置信度小于阈值 minconf=0.6，所以不能输出。因为不能从 cdf 获得 3-项形式的后件，所以 ap-genrules 过程终止，因为 F_4 为空，所以算法 4.2 也终止。

表 4.3 从频繁 3-项集产生关联规则

1-项后件			2-项后件		
序号	规则	置信度	序号	规则	置信度
9	cd⇒f	0.60	12	f⇒cd	0.43
10	cf⇒d	0.75	13	d⇒cf	0.33
11	df⇒c	0.75	14	c⇒df	0.43

4.4.1.2 序列模式挖掘

本节我们将描述 AprioriAll 算法在表 4.1 的数据集上查找频繁最大顺序模式的情形，取阈值 minsup=0.3。图 4.3 简约描述了前三个步骤，即排序阶段、频繁项集阶段和转换阶段。在排序阶段，将数据集里的事务用<主键 SID，次键 TT>来排序。在频繁项集阶段，用 Apriori 算法类似的方法来获得频繁项集，此处要注意，频繁项集的支持度是包含它的事务队列的数量，而不是包含它的事务的数量。所以在本例中，频繁 1-项集为{c,d,f}。在转换阶段，每个事务序列被转换为如图 4.3 所示的频繁项集的集合的列表，即将序列中的事务替换成该事务包含的频繁项集的集合的列表。请注意，事务序列 4 的第 2 个事务(a)被丢掉了，因为它只仅包含一个非频繁项集{a}。

AprioriAll 通过调用函数 apriori-gen-2 从 F_1 生成候选序列 C_2，如表 4.4 所示。apriori-gen-2 和 apriori-gen 类似，只是采用的链接操作不同：apriori-gen-2 的链接操作从两个(k−1)-序列生成两个新的 K-序列，而 apriori-gen 从两个 k−1 项集只能生成一个 K-项集，

初始数据库

TID	SID	TT	Items
001	1	May 03	c d
002	1	May 05	f
003	4	May 05	a c
004	3	May 05	c d f
005	2	May 05	b c f
006	3	May 06	d f g
007	4	May 06	a
008	4	May 07	a c d
009	3	May 08	c d f g
010	1	May 08	d e
011	2	May 08	b d
012	3	May 09	d g
013	1	May 09	e f
014	3	May 10	c d f

已排序数据库

SID	TID	TT	Items
1	001	May 03	c d
1	002	May 05	f
1	010	May 08	d e
1	013	May 09	e f
2	005	May 05	b c f
2	011	May 08	b d
3	004	May 05	c d f
3	006	May 06	d f g
3	009	May 08	c d f g
3	012	May 09	d g
3	014	May 10	c d f
4	003	May 05	a c
4	007	May 06	a
4	008	May 07	a c d

频繁项集

Fitemset	Mapped To
(c)	1
(d)	2
(c d)	3
(f)	4
(c f)	5

转换后的数据库

SID	Sequence	Transformed sequence	After mapping
1	<(c d)(f)(d e)(e f)>	<{(c),(d),(c d)}{(f)}{(d)}{(f)}>	<{1,2,3}{4}{2}{4}>
2	<(b c f)(b d)>	<{(c),(f),(c f)}{(d)}>	<{1,4,5}{2}>
3	<(c d f)(d f g)(c d f g)(d g)(c d f)>	<{(c),(d),(cd),(f),(c f)}{(d),(f)}{(c),(d),(cd),(f),(c f)}{(d)}{(c),(d),(cd),(f),(c f)}>	<{1,2,3,4,5}{2,4}{1,2,3,4,5}{2}{1,2,3,4,5}>
4	<(a c)(a)(a c d)>	<{(c)}{(c),(d),(cd)}>	<{1}{1,2,3}>

图 4.3　从初始数据库到转换后数据库的过程

例如，在求 C_2 时，链接<1>和<2>生成了<12>和<21>。此外，用同一个<1>链接可以生成<11>，这样做是必要的，因为某个频繁项集在序列中多次出现是可以的。

计算每个候选序列的支持度和 Apriori 算法是类似的，也要用哈希树。表 4.4 显示了频繁 2-序列的集合 F_2 的求解，进而 F_2 被用于生成候选序列集合 C_3。注意链接<11>和<12>生成的是<112>而非<121>，因为其序列的<21>不包含在 F_2 中。反复执行候选生成和支持计算，直到不再有频繁序列生成。本例中，从 F_4 中不再能生成候选 5-序列，所以 F_5 也是空，迭代到此终止。最后，AprioriAll 输出<1424>，<2424>，<3424>，<11>，<13>和<52>作为最大频繁序列，其他频繁序列都包含在这其中的某一个里了。

表 4.4　频繁序列和候选序列

F_1	C_2	F_2	C_3	F_3	C_4	F_4
<1>	<11><12><21><13>	<11><12>	<111><112><113>	<124><142>	<1244>	<1424>
<2>	<31><14><41><15>	<13><14>	<114><122><132>	<144><224>	<1424>	<2424>
<3>	<51><22><23><32>	<22><32>	<124><142><134>	<242><324>	<2244>	<3424>
<4>	<24><42><25><52>	<24><42>	<144><222><224>	<342><244>	<2424>	
<5>	<33><34><43><35>	<52><34>	<242><322><324>	<424><344>	<3244>	
	<53><44><45><54>	<44>	<342><244><422>		<3424>	
	<55>		<424><442><522>			
			<344><444>			

4.4.2 性能评估

在本节中，我们将讨论 Aprorid 算法的性能，通过设置不同的 minsup、minconf 和事务数量，考察算法运行时间以及生成的关联规则、频繁项集的数量。我们用 Christian Borgelt [1]的算法实现来进行性能评估，这个软件提供了一些选项用以逼真地模拟经典的 Apriori 算法。我们关闭了用于过滤事务中的无用项的项排序功能。

我们使用 UCI 机器学习资料库[8]中的 Mushroom 数据集作为测试数据，它包含了 8124 个实例，用 23 个标称型属性(其中有一个是类别属性)描述。将每个实例当做一个事务，将每个实例的属性名和属性值连起来当做一个项，例如“cap-shape＝x”中 cap-shape 是属性名，而 x 是该属性的取值。数据集里有 2480 个实例缺少一个属性值，所以对应的事务有 22 个项，而其他事务则有 23 个项。有些属性值在不同的属性中表示的含义不同，例如，属性值“n”在“气味”属性中是“没有”的意思，而在“帽子颜色”这个属性中是“棕色”的意思。最终得到不重复的有效＜属性名-属性值＞对的数量是 118。

首先，我们考察在各种 minsup、minconf 设置下 Apriori 算法的运行时间，参见图 4.4。在本节中的所有运行时间测量的平台都是运行 Windows XP 的 PC，配置 2.8GHz 奔腾 IV 的 CPU 和 4GB 的内存。实验中将每条规则包含的项的数量上限设为 5(为了方便)，下限设为 2(确保规则的前件存在)。此外，没用前缀树存储事务。实验结果显示 minconf 的变化对运行时间影响不大，但 minsup 变小时运行时间会指数倍地增长。图 4.5 显示，关联规则数量与运行时间的情形类似，这是因为随着 minsup 变小，频繁项集的数量呈指数倍地增长，如图 4.6 所示。这些结果表明参数 minsup(或说项集的反单调性)对于排除非频繁项集是极为有效的。

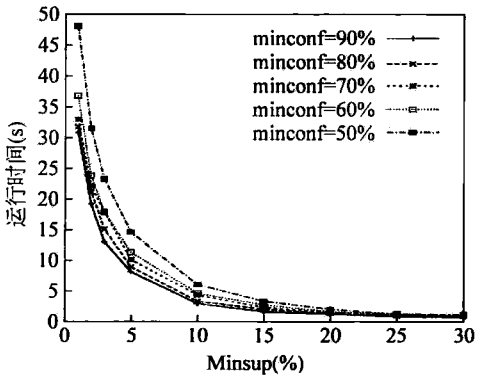


图 4.4 不同 minsup 和 minconf 值的运行时间

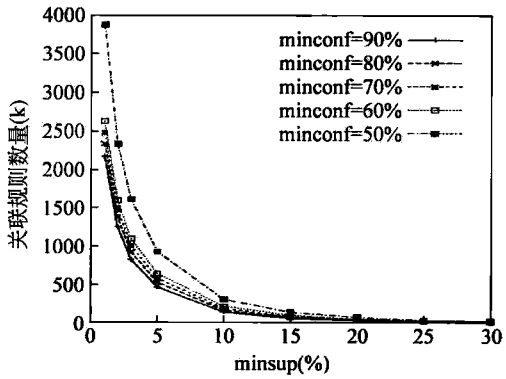


图 4.5 根据 minsup 值和 minconf 值变化的关联规则数量曲线

接下来，我们展示运行时间与事务数量间的关系，如图 4.7 所示。在评估测中，我们通过复制原始数据集来生成更大的数据集(最大达 4 倍)，由于每个项在各数据集里的比重是一样的，所以这些数据集产生的关联规则和频繁项集的数量是一样的。实验结果显示，运行时间随着事务数量的增加而线性增加。因此，在项的分布一定的情况下，Apriori 运行时间主要是由 minsup 决定的，minconf 和事务数量所起的作用相对小得多。

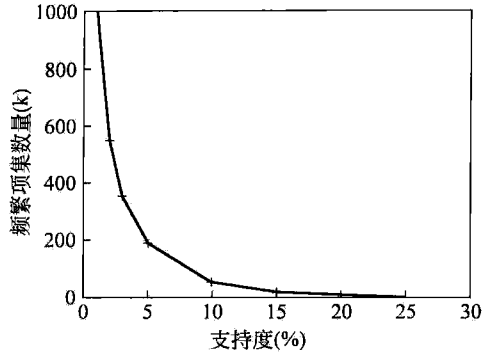


图 4.6 不同 minsup 值对应的频繁集关系图

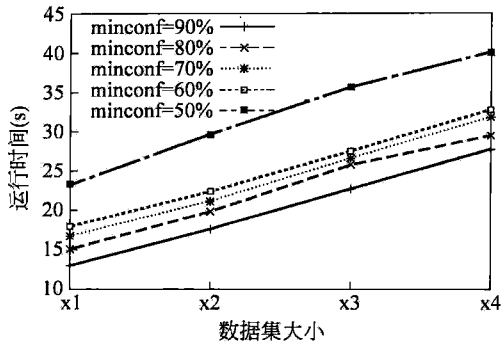


图 4.7 不同大小的数据集的运行时间关系表(minsup=5)

最后,在通过实验简要地说一下关联规则挖掘,为方便起见,实验中的类别属性仅用一个项表示。类别的值可取"可食用"(e)或者"有毒"(p)。在 minsup=0.3,minconf=0.9 的情况下,可以得到一个典型的规则:"odor=n gill-size=b ring-number=o⇒class=e"。这条规则是说如果蘑菇没有气味且菌鳃较宽且仅有一个环,那么这种蘑菇石是可食用的。在所有后件为 class=e 的规则中这是最简单的一条,其置信度为 1,支持度达到最大值 0.331。在本例中,我们还用 Weka 中的学习算法 J48(取默认设置)得到了一棵决策树,"气味"和"菌鳃大小"的两个属性分别处于第一和第三个测试节点。从该决策树可以得到与上一个规则类似的规则"odor=n spore-print-color=w gill-size=b⇒class=e",这条规则的置信度高达 1.0,但数据集中只有 528 个实例符合它,而符合前一条关联规则的实例高达 2689 个。从另一角度讲,minsup 设得过高导致无法找到置信度为 1.0,后件为"class=p"的规则。当取 minsup=0.2,这样的规则就会被发现。

一般来说,如果 minsup 较高,我们可以用较短的运行时间获得少量的关联规则,但其中许多规则可能是平凡的。为了找到更有意义的规则,我们可以取较小的 minsup,但这可能导致不可接受的运行时间和数量庞大的关联规则,这种情况下想找到有意义的规则是很困难的。所以,频繁项集和关联规则的挖掘任务要求更高效的算法和更好的度量方法。我们在 4.5 节中讨论这些主题。

4.5 高级主题

自从 Agrawal 和 Srikant 提出第一个频繁模式和关联规则挖掘算法以来,涌现出了很多种改进、扩展和应用,涵盖了面向大规模数据集的高效挖掘方法、多样数据类型的处理、新型挖掘任务的扩展,乃至各式各样的新应用。本节对若干重要的高级主题进行简单地探讨。Han 等[16]和 Goethals[15]撰写的频繁模式挖掘的综述及其引用的大量参考文献都是非常有价值的。

4.5.1 改进 Apriori 类型的频繁模式挖掘

在 Apriori 算法的框架下,可以引入很多技术来改进频繁项集挖掘算法的效率。这些

技术主要包括哈希、分治、采样和垂直数据格式等。

哈希技术可以用来缩减候选项集的数量。每个项集被一个合适的哈希函数散列到一个相应的桶中。因为一个桶能够包含不同的项集，如果一个项集的计数小于最小支持度，就将其从这个桶中移除掉。DHP[26]就是这个思想。

划分是指将一个整体的挖掘问题转化为 n 个更小的问题[29]。这里是指将数据集分成互不重叠的 n 部分，每部分都能被放入内存，这样就可以对每部分数据分别进行挖掘。任何可能的频繁项集必定至少在一个部分上表现为频繁项集，按照这一思路找到的频繁项集都是候选项集，检验工作仅需遍历整个数据集一次。

采样是指从整个数据集里随机选出的数据子集，在该子集上进行挖掘。由于这种方法不能确保找到所有的频繁项集，所以实践中通常选取较低的支持度阈值。使用这种方法必须对精度和效率进行权衡。

垂直数据格式是指将 TID 与每个项集关联。Apriori 使用水平数据格式，即将频繁项集关联到每个事务上。如果使用垂直数据格式，挖掘中就能对 TID 集合进行求交集的运算。项集的支持度计数就是 TID 集合的长度，(采用垂直数据格式表示时)该长度可以直接得到，这样就不用扫描整个数据集。这个技术要求，对于给定的候选项集的集合，其 TID 必须能全部放入内存，但内存一般是放不下它们的。对此，可以使用深度优先搜索来显著地缩减这一规模，Eclat[43]就是使用这种策略。在深度优先的方法中，内存中至多需要存储具有相同的前 k−1 项(k−1 前缀)的那些 k-项集的 TID 列表，搜索深度为 d(k≤d)。

4.5.2 无候选的频繁模式挖掘

对 Apriori 算法最出色改进要数 FP-growth(频繁模式增长)算法。该算法根本不用生成候选项集[17,18]，而是采用“分而治之”的策略：(1)压缩数据集并将频繁项放入 FP-tree(频繁模式树)，它保留了所有必要的信息；(2)将压缩数据集分成多个条件数据集，每个条件数据集关联到一个频繁项集上并分开来对其进行挖掘。该算法只需扫描数据库两次：第一次扫描，得出所有频繁项及其支持度计数(即频率)，并在每个事务中将频繁项依据支持计数进行降序排列；第二次扫描，将每个事务的项都合并进 FP-tree，对在不同事务中出现的公共项(即节点)进行计数。每个节点与一个项及其计数相关联，具有相同标签的节点被称为 node-link 的指针链接在一起。因为项是以按频率降序排列的，FP-tree 上接近根节点的那些节点能被更多的事务共享，这样就将所有必要信息以一种非常紧凑方式表示了。模式增长算法在 FP-tree 上运行，按频率递增的顺序选定一个项并提取出包含该项的频繁项集，具体说就是模式增长算法在条件 FP-tree(条件数据集)上递归调用自身，就是说 FP-tree 是项的条件。FP-growth 速度比 Apriori 算法快一个数量级。FP-growth 算法请参加算法 4.5。用 F[∅](FP-tree)可返回所有的频繁项集。可以很容易看出，Han 等人提出的分而治之策略相当于没有候选项集的深度优先搜索。称 D^i 为 i-projected 数据集，它通常比 FP-tree 表示的整个数据集小很多，可以放进内存(整个数据集一般是不能放入内存)。模式增长这一思想也体现在闭合项集挖掘[27](请见 4.5.4 节)和顺序模式挖掘[28](请见 4.5.8 节)中。

算法 4.5 FP-Growth Algorithm: F[I](FP-tree)

```
F[I]=∅;
foreach i∈Ithat is in D in frequency increasing order do begin
    F[I]=F[I]∪{I∪{i}};
    D^i=∅;
    H=∅;
    foreach j∈Iin Dsuch that j<i do begin
        //(j is more frequent that i)
        Select j for which support (I∪{i,j})≥minsup;
        H=H∪{j};
    end
    foreach(Tid,X)∈D with i∈ X do
        D^i=D^i∪{(Tid,{X\{i}}∩H)};
    Construct conditional FP—tree from D^i;
    Call F[I]∪{i}](conditional FP-tree);
    F[I]=F[I]∪F[I∪{i}](conditional FP-tree);
end
```

4.5.3 增量式方法

有些情况下，数据集不是固定的，新的事务会不断地加进来，这时一些本来频繁的项可能变得不再频繁了(失败者)，而一些本来不频繁的项变得频繁起来了(成功者)。每当数据集发生更新时，Apriori 或者其他频繁模式挖掘算法就需要被重新运行，而这样做效率很低。文献[12]中提出的 FUP 算法能在 Apriori 算法的框架下实现对频繁项集的增量式更新。由于增量数据集 ΔD 通常比原始的数据集 D 小很多，所以该算法的处理效率很高。

记 F_k，F'_k 分别是 D 和 $D \cup \Delta D$ 中的频繁 k-项集，C_k 是 $D \cup \Delta D$ 中的候选频繁项集。在第 k 次迭代中，apriori-gen 函数从 F'_{k-1} 生成 C_k。首先，考察 F_k 中的项集：(a) F_k 中的任何项集，如果它包含大小为 k−1 的失败者(在 F_{k-1} 中而不在 F'_{k-1} 中)，则无需再检查 ΔD 即可将该项集从 F_k 中滤除；(b) 除此之外的 F_k 中的其他项集要在 ΔD 上进行计数，从而识别出 $D \cup \Delta D$ 上的频繁项集(记之为 A)，并将它们从 C_k 中排除因为我们已经知道它们是频繁的了。接着，再考察不在 F_k 中的项集：(a) 将它们在 ΔD 上进行计数，把那些在 ΔD 上不频繁的项集从 C_k 中移除，因为我们知道它们在 D 中是不频繁的；(b) 将 C_k 中剩下的项集在 $D \cup \Delta D$ 上进行计数，从而识别出那些频繁的项集(记之为 B)。F'_k 就是 A∪B。从上面可以看出，虽然对每个 k 算法 FUP 都必须扫描更新数据集 $D \cup \Delta D$，但 C_k 其实都是很小的，所以能保证高效。实验表明，在有 2%～5%增量的更新数据集 $D \cup \Delta D$ 上重新运行 Apriori 算法，FUP 要比它快 2～16 倍。

4.5.4 稠密表示：闭合模式和最大模式

说一个项集(模式)X 是一个最大项集，是指不存在另一个项集 X′是 X 的真超集。说一个项集(模式)X 是一个闭合项集，是指(1)不存在另一个项集 X′是 X 的真超集，并且(2)每个包含 X 的事务也必定包含 X′。如果它们的支持度不少于最小支持度，那么就是频繁的。闭合项集具有性质 $I(T(X))=X$，此处 $T(X)=\{t \in D \mid X \subseteq t\}$；$I(S)=\bigcap_{t \in S} t$；$S \subseteq D$。对于任

何两个项集 X 和 Y，如 X⊂Y 并且它们的支持度相同，则 X 必定不是闭合项集。闭合项集是无损的，而最大项集却不一定，所以，只有找到了闭合项集，就能导出所有频繁项集。说一个规则 X⇒Y 是频繁闭合项集上的一个关联规则，是指(1)X 和 X∪Y 都是频繁闭合项集，同时(2)不存在一个频繁闭合项集 Z，使得 X⊂Z⊂(X∪Y)成立，以及(3)该规则的置信度不低于 minconf。只要找到了频繁闭合项集，就可以生成关联规则的完整集合。

CLOSET 算法对数据集进行划分并把原始挖掘问题分解形成子问题的集合，每个子问题有一个与之对应的条件数据集，人们已经知道这是一种高效的方法 [27]。首先，导出所有频繁项，并按支持度降序排序成 f_list=<i_1,i_2,…,i_n>。第 j(1≤j≤n)个子问题，是要找到包含 i_{n+1-j}但没有 i_k(n+1−j<k≤n)的全部频繁闭合项集。条件数据集 i_{n+1-j}是包含项 i_{n+1-j}的事务的子集，并且舍弃了 f_list 中所有的非频繁项、项 i_{n+1-j}及其后的项。然后，生成对应的 FP-tree 并用于搜索。如有必要，每个子问题都可以被递归分解。要将频繁闭合项集从条件数据集里识别出来，需要用到一些性质：(1)如果 X 是一个频繁闭合项集，就不再有(X 之外的)项出现在 X-条件数据集的每个事务中；(2)如果项集 Y 是出现在 X-条件数据集里的每个事务中的最大项集，并且 X∪Y 还未被一些已知的具有相同支持度的频繁闭合项集包含，则 X∪Y 就是一个频繁闭合项集。如 FP-growth 算法，还可对其他一些方面进行了优化。

另一个算法 LCM 是已知最有效的闭合模式(项集)发现算法[34]。该算法通过闭包运算导出频繁闭合项集，并且不需要生成非闭合项集。将项集 X 的闭包记为 Clo(X)，表示包含 X 的唯一最小闭集，也就是 I(T(X))。不失一般性，我们将事务数据集里的所有项用连续的自然数进行索引。这样，称 X(i)=X∩{1,…,i}为 X 的 i-前缀，就是索引不大于 i 的那些元素构成的子集。将闭合项集 X 的核心索引记为 Core_i(X)，就是满足 T(X(i)=T(X)的最小索引。LCM 算法会从频繁闭合项集 X 生成频繁闭合项集 Y，其中 Y=Core_i(X∪{i})；X(i−1)=Y(i−1)，i 是满足 i∉X 和 i>Core_i(X)的项。Y 被称为 X 的保留前缀的闭包扩张(或简称 ppc-扩张)。LCM 算法以深度优先的方式，从空项集开始递归地执行这种闭包运算来生成更大的项集。LCM 算法的闭合项集枚举算法是完整和非冗余的，这是因为：如果 Y 是一个非空闭合项集，那么仅存在一个闭合项集 X，使得 Y 是 X 的 ppc-扩张。因为 LCM 算法是从 T(X)和 I 的一个子集来生成一个新的频繁闭合项集 Y 的，所以为 X 枚举出所有频繁闭合项集的时间复杂度是 O(||T(X)||×|I|)，此处||T(X)||是 T(X)中所有事务的大小的总和。令 C 为 D 中所有频繁闭合项集的集合，那么，LCM 算法具有关于|C|的线性时间复杂度，线性因子取决于||T||×|I|。事实上，为了改进计算时间和内存用量，LCM 用了三种技巧：事件交付、数据集随时缩减和快速前缀保留测试。事件交付是指仅对 T(X)扫描一次就能为所有 i 构建出 T(X∪{i})，而不是为每个 i 都执行一遍扫描。数据集随时缩减是指，在执行当前闭合项集的迭代处理前，从数据集里除去不必要的事务和项，从而降低了计算时间和内存用量。快速前缀保留测试是指，在执行 ppc-扩张时，只检查满足 j<i,j∉X(i−1)的那些项 j，它们都包含在 T(X∪{i})中的最小事务里而不需要真的生成闭包，这样就显著减少测试等式 X(i−1)=Y(i−1)所需访问的项的数量。如果一个项 j 包含在 T(X∪{i})的每一个事务中，那它就包括在 Clo(X∪{i})中，那么就有 X(i−1)≠Y(i−1)。

4.5.5 量化的关联规则

如果项的取值是连续型的，那经典的频繁项集挖掘算法不适用了，必须要先将这些连续

值离散化而且要确定恰当的区间，这就是所谓的量化频繁项集(QFI)挖掘。项的取值可以是类别型也可以是数值型，例如，项集{<年龄:[30,39]>,<房主:是>,<结婚:是>}，项的描述为<属性:取值(定义域)>。QFI 挖掘是最早提出的量化关联规则挖掘[31]算法，不过真正被广泛使用的是后来提出的基于密度的子空间聚类算法。QFI 可以被看作是数值属性空间中包含事务聚类的轴平型超矩形，SUBCLUE[20] 和 QFIMiner [36]是两个这种类型的算法。QFIMiner 可以在数值属性和分类属性构成的所有子空间中找到所有支持度不低于 minsup 的密集聚类，该算法的时间复杂度是 O(N logN)，N 是事务的数量。每个频繁项集的每个项的最优取值区间是用一个类似于 Apriori 的分层处理算法确定的，该算法的关键是要用到稠密聚类的反单调性。QFIMiner 比 SUBCLUE 更快而且可扩展性更好。

4.5.6 其他的重要性/兴趣度度量方法

支持度-置信度框架的问题是没有确定 minsup 和 minconf 合适取值的有效方法。minsup 如果设置过高则会丢失重要的规则，如果设置过低又会产生太多的规则。事实上，对于有些应用，可能一条具有非频繁项集的规则反而是非常有意义的。还有，这一框架不能有效地刻画关联性，例如某个规则 X⇒Y 能同时满足 minsup 约束和 minconf 约束，但 X 和 Y 之间其实不相关，即 support(X)×support(Y)=support(X∪Y)。

因此，人们想到对规则的重要性或兴趣度进行度量，按照度量的得分来选取规则，这些度量措施提升(lift)、杠杆(leverage)、冗余(redundancy)、产出(productivity)以及一些著名的统计量如卡方、相关系数和信息增益等。这里仍然可以讲支持度和置信度(只需将 minsup 和 minconf 设为 0，即可让它们不起作用)。

提升和杠杆分别表示规则支持度与 XY 独立情况下支持度之间的比值和差值。这两种度量方法目的都是发现在 X 和 Y 间具有很强关联性的规则。

$$\text{lift}(X\Rightarrow Y)=\frac{\text{confidence}(X\Rightarrow Y)}{\text{confidence}(\varnothing\Rightarrow Y)}=\frac{\text{support}(X\Rightarrow Y)}{\text{support}(X)\times\text{support}(Y)}$$

$$\begin{aligned}\text{leverage}(X\Rightarrow Y)&=\text{support}(X\Rightarrow Y)-\text{support}(X)\times\text{support}(Y)\\&=\text{support}(X)\times(\text{confidence}(X\Rightarrow Y)-\text{support}(Y))\end{aligned}$$

冗余(redundancy)是指，删除掉满足 $\exists Z\in X:\text{support}(X\Rightarrow Y)=\text{support}(X-[Z]\Rightarrow Y)$ 的规则 X⇒Y。产出(productivity)是比冗余更强有力，它要求一条规则的推进(improvement)必须大于 0，推进的定义如下：

$$\text{improvement}(X\Rightarrow Y)=\text{confidence}(X\Rightarrow Y)-\max_{Z\subset X}(\text{confidence}(Z\Rightarrow Y))$$

由于冗余规则的推进不可能大于 0，所以产出约束将会删除掉所有冗余规则，此外，如果给定前件中的保留项，产出约束还能删除掉前件和后件独立的规则。

统计度量对于寻找判别模式(项集)是很有用的。然而，由于上述统计度量方法不具备反单调性，所以寻找最佳 k 个模式(或规则)是个困难任务。但是，这些统计度量方法关于自身参数是凸的，那么，就可以估计出与给定结论 Y(通常是某个类别值)对应的模式 X 的超集的上界[23]，这样就能缩减搜索空间。

Webb 的 KORD 算法[39]在 XY(Y 不固定)构成的整个空间上搜索 k-最优规则，并且利用杠杆来度量各种约减策略的优化效果。

4.5.7 类别关联规则

如果事务 t 是与类别 cl 关联的，那么将关联规则用于分类就是很自然的事了。面向分类任务的关联规则称为类别关联规则(CAR)，其形式为$\{<p_1:q_1>,<p_2:q_2>,\cdots,<p_m:q_m>\}\Rightarrow cl$。这里，数值项取区间值，类别项取类别值。CBA[22]、CMAR[21]和 CAEP [14]是代表性的基于 CAR 的分类系统，特别是 CAEP 引入了紧急模式的概念并使用了所有类关联规则的强度。记$\mathcal{D}_{cl}$为 D 中 cl 类的所有实例，记$\mathcal{D}_{cl}$上的一个项集 a 的支持度为$\mathcal{D}_{cl}(a)=|\{t\in\mathcal{D}_{cl}|a\in t\}|/|\mathcal{D}_{cl}|$。对每个类别 cl，从$\mathcal{D}_{cl}$可以导出 QFIs，FQFI(cl)的集合，在里面的每一个项集 a 都满足 $\text{support}_{\mathcal{D}_{cl}}(a)\geqslant\text{minsup}$。进而，对每个 $a\in\text{FQFI}(cl)$，为每个类别定义增长率 $\text{growth\_rate}_{\bar{\mathcal{D}}_{cl}\to\mathcal{D}_{cl}}(a)=\text{support}_{\mathcal{D}_{cl}}(a)/\text{support}_{\bar{\mathcal{D}}_{cl}}(a)$，$\bar{\mathcal{D}}_{cl}=\mathcal{D}-\mathcal{D}_{cl}$表示 cl 类的对立集里的实例。当 a 的增长率不低于阈值 $\rho(\geqslant 1)$时，即 $\text{growth\_rate}_{\bar{\mathcal{D}}_{cl}\to\mathcal{D}_{cl}}(a)\geqslant\rho$，就称 a 为紧急模式(EP)并且生成规则 a⇒cl(规则头是 a，规则体是 cl)，其背后的思想是力图让选择的规则体足以将 cl 与其他类别区分开来。再记 FEP(cl)为基于这一度量从 FQFI(cl)中选出的 EP，一个 EP 的强度是 $\text{support}_{\mathcal{D}_{cl}}(a)$和 $\text{support}_{\bar{\mathcal{D}}_{cl}}(a)$之间的相对差定义为：

$$\text{support}_{\mathcal{D}_{cl}}(a)/(\text{support}_{\mathcal{D}_{cl}}(a)+\text{support}_{\bar{\mathcal{D}}_{cl}}(a))$$
$$=\text{growth\_rate}_{\bar{\mathcal{D}}_{cl}\to\mathcal{D}_{cl}}(a)/(\text{growth\_rate}_{\bar{\mathcal{D}}_{cl}\to\mathcal{D}_{cl}}(a)+1)$$

进而可以定义聚合分数 $\text{score}(t,cl)=\sum_{a\subseteq t,a\in\text{FEP}(cl)}\frac{\text{growth\_rate}(a)}{\text{growth\_rate}(a)+1}*\text{support}_{\mathcal{D}_{cl}}(a)$，它代表根据 FEP(cl)中的 EP，t 被分到类别 cl 中的可能性。因为 EP 的数量在类别 cl 上分布不均匀，有些类的实例可能会得分较高。为了抵消这种偏差，需引入另外一个因子，称为基础分数，即$\{\text{score}(t,cl)|t\in\mathcal{D}_{cl}\}$中所有聚合分数的中值，这样可以得到归一化的分数 $\text{norm\_score}(t,cl)=\frac{\text{score}(t,cl)}{\text{base\_score}(cl)}$。得到最高归一化分值的类别 cl 被分配 t。这样做得到的效果很好。

CAEP 算法的一个问题是，它用熵度量方法独立离散化每一个数值属性而不考虑多个属性之间的依赖。因此，可能会导致同一类别的实例簇碎片化。一种很自然的解决方案是将 QFIMiner 和 CAEP 结合，那就是 LSC-CAEP 算法[37,36]。

4.5.8 使用更丰富的形式：序列、树和图

频繁项集挖掘最初是在简单的事务数据集上做的，但后来被推广到序列、树和图等丰富的表达形式上。在 4.2.2 节里，已经介绍了 Agrawal 和 Srikant 关于序列模式挖掘的开创性工作。PrefixSpan [28] 是另一个代表性的频繁序列模式挖掘算法，这种模式增长型算法采取和 FP-growth 相似的分而治之策略，从而避免了在寻找更大模式中对较小候选的无用的枚举。PrefixSpan 首先找到仅含一个项的序列模式，并对每个项 i_k 抽取出包含该项的序列的集合，即$<i_k>$-投影数据集。然后，PrefixSpan 从每个投影数据集上发现以$<i_k>$为前缀的长度为 2 的频繁序列模式，并为每个新发现的长度为 2 的模式生成一个投影数据集，用以发现长度为 3 的序列模式。这一过程是重复进行，直到没有更多的序列模式被发现。

树是由顶点集 V 和边集 E 定义的，标签树是将一组标签赋给顶点和/或边，一条边将一

个顶点与另一个顶点连接在一起。在树中，任意每两个顶点之间存在一条或多条通路，但不能有环。TreeMinerV [44] 和 FREQT [7] 是两个从树集合中挖掘频繁子树的典型算法。这两个算法是独立出现的，但它们有相同的频繁子树分层枚举策略，即为了从 k-子树生成有 k+1 个顶点((k+1)-子树)的频繁子树，要在 k-子树的最右路径的所有可能位置上加一条边，这条边的另一端对应于 k-子树中的某个顶点。Dryade[33]是一种可以得到频繁闭合子树的树挖掘算法。闭合子树是指具有相同频率的子树中的最大子树。称深度为 1 的闭合子树为“砖”，Dryade 算法以“砖”为最基础层次来逐层组装生成频繁闭合子树，这是其独有的特色。

图是树的超类，可以有环。AGM[19]是第一个通过完全搜索从图集合中挖掘频繁子图的算法。该算法以 Apriori 为基础，用两个共享同一(k－2)-子图的已知的频繁(k－1)-子图生成一大小为 k 的候选子图(k-子图)。由于不知道两个第(k－1)顶点之间的边的情况，因此各种可能都要考虑。AGM 从一对(k－1)-子图生成两个 k-子图，其中一个 k-子图的两个(k－1)-子图之间有一条边，另一个没有边(边上没有定义标签就是这种情况)。基于 Apriori 的方法能对频繁子图进行系统完整的搜索，但它会产生大量在给定图集合中并不存在的候选子图。AGM 用邻接矩阵来表示图，并引入了一种规范形式来求解子图同构的问题，但这是一个 NP 完全问题。gSpan[41]是一种典型的基于模式增长的子图挖掘算法，它在寻找频繁子图过程中采用深度优先方式，在已知频繁个子图的最右路径上的每个可能的位置加一条边。gSpan 只考虑给定图集里已存在的边，所以不会生成图集里不存在的候选子图。GBI [42]和 SUBDUE[13]采取贪婪算法寻找频繁子图，以递归的方式将一个典型子图在图中的每一次出现替换为一个新顶点。典型度的概念定义在基于频率的度量方法之上，比如 GBI 用信息增益，SUBDUE 用最小描述长度。DT-ClGBI[25]能从已知类别的图训练集生成一个决策树，进而分类类别未知的图。该算法要在决策树的每一个测试节点上调用图形挖掘算法 Cl-GBI [24](GBI 算法的扩展)，得到的频繁子图被用作为图的属性，选择具有最大判别能力的频繁子图将图集分裂成两个分支：包含该子图的图与不包含该子图的图。

4.6 小 结

对于从事数据挖掘的人来说，用 Apriori 系列的算法进行实验是第一件应该做的事情。本章首先介绍了相关的基本概念和 Apriori 算法族(Apriori、AprioriTid、AprioriAll)，然后用一些示例描述了这些算法的工作机制，接着用一个典型的免费 Apriori 实现对该算法进行性能评估实验。因为 Apriori 算法非常基本也容易实现，所以产生了许多变体。本章还讨论了 Apriori 方法的一些局限性，并对近年来频繁模式挖掘方法的重要发展做了综述。当然，有些主题在本章中没有谈到，例如限制的使用、巨大模式、噪声处理和 top-k 代表等。

4.7 习 题

1. 请证明 Apriori 算法可以导出给定事务集的全部频繁项集。
2. 请证明以下关系：

$$\text{support}(X \cup Y \cup Z) \geqslant \text{support}(X \cup Y) + \text{support}(X \cup Z) - \text{support}(X)$$

此处 X、Y、Z 都是数据集里的频繁项集。

3. 给定表 4.5 所示的数据集，用 Apriori 算法和 AprioriTid 算法在 minsup = 0.3 的设置下找出所有频繁项集并比较两种算法的效率。

表 4.5 习题 3 所用数据集

TID	项 集	TID	项 集
T01	Cheese, Milk, Egg	T06	Apple, Chese, Egg, Orange
T02	Apple, Cheese	T07	Bread, Cheese, Orange
T03	Apple, Bread, Cheese, Orange, Grape	T08	Cheese, Egg, Grape
T04	Bread, Egg, Orange	T09	Bread, Cheese, Egg, Grape
T05	Cheese, Milk, Grape	T10	Bread, Cheese, Grape

4. 请解释 hash-tree 和 trie 的关系。

5. 请为表 4.5 画出一棵 FP-tree 并解释如何从 FP-tree 中导出频繁项集。

6. 下载 Weka 并将其安装到你的机器上，用其自带的 Soybean 数据集运行 Apriori 算法来挖掘关联规则，请使用多种不同的度量和相同的最小阈值(固定其他参数)。然后，报告不同度量引起的关联规则的变化。

7. 请参考文献[10]为 4.4 节的数据集画出前缀树(prefix tree)，并解释为何该方法能改善频率计数的性能。

8. 在 FP-tree 中，事务的项是以支持计数的降序排列的，但在 prefix tree 中缺失以支持计数的升序排列的。请讨论两种数据结构为什么对项采用不同的排列顺序？

9. 当一个事务数据集有少量极长事务时，基于 Apriori 的频繁项集挖掘算法会耗时极长。请解释其原因并提出有效的解决方法。

10. 给定如表 4.6 所示的序列数据集，请在 minsup = 0.5 的设置下用 AprioriAll 算法找到所有频繁序列模式。

表 4.6 习题 10 所用序列数据集

SID	处 理 顺 序	SID	处 理 顺 序
S01	<(bc)(d)(ab)(def)>	S03	<(cef)(df)(ab)(f)>
S02	<(abc)(cf)(df)>	S04	<(be)(ac)(cdf)>

参 考 文 献

[1] http://www.borgelt.net/apriori.html.

[2] http://www.cs.bme.hu/~bodon/en/apriori/.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th International Conference on Very Large Data Bases* (*VLDB* 1994), pages 487 499, 1994.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. IBM Research Report

RJ9839, IBM Research Division, Almaden Research Center, 1994.

[5] R. Agrawal and R. Srikant. Mining sequential patterns. IBM Research Report RJ9910, IBM Research Division, Almaden Research Center, 1994.

[6] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of the 11th International Conference on Data Engineering* (*ICDE* 1995), pages 3 14, 1995.

[7] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, and S. Arikawa. Efficient substructure discovery from large semi-structured data. In *Proc. of the 2nd SIAM International Conference on Data Mining*, pages 158 174, 2002.

[8] C. Blake and C. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[9] F. Bodon. Surprising results of trie-based fim algorithms. In *Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations* (*FIMI'04*), volume 126 of CEUR Workshop Proceedings, 2004. http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-126/bodon.pdf.

[10] C. Borgelt. Efficient implementations of Apriori and Eclat. In *Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations* (*FIMI'03*), volume 90 of *CEUR Workshop Proceedings*, 2003. http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-90/borgelt.pdf.

[11] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proc. of ACM SIGMOD International Conference on Management of Data* (*SIGMOD* 1997), pages 255 264, 1997.

[12] D. W. Cheung, J. Han, and C. Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. In *Proc. of the* 1996 *ACM SIGMOD International Conference on Management of Data*, pages13 23, 1996.

[13] D. J. Cook and L. B. Holder. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, Vol. 1, pages 231 255, 1994.

[14] G. Dong, X. Zhang, L. Wong, and J. Li. Caep: Classification by aggregating emerging patterns. In *Proc. of the 2nd International Conference on Discovery Science* (*DS'99*), *LNAI* 1721, Springer, pages 30 42, 1999.

[15] B. Goethals. Survey on frequent pattern mining, 2003. http://www.adrem.ua.ac.be/bibrem/pubs/fpm survey.pdf.

[16] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: Current status and future direction. *Data Mining and Knowledge Discovery*, Vol. 15, No. 1, pages 55 86, 2007.

[17] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. of the* 2000 *ACM SIGMOD International Conference on Management of Data*, pages 1 12, 2000.

[18] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, Vol. 8, No. 1, pages 53 87, 2004.

[19] A. Inokuchi, T. Washio, and H. Motoda. General framework for mining frequent subgraphs from labeled graphs. *Fundamenta Informaticae*, Vol. 66, No. 1-2, pages 53 82, 2005.

[20] K. Kailing, H. Kriegel, and P. Kroger. Density-connected subspace clustering for high-dimensional data. In *Proc. of the 4th SIAM International Conference on Data Mining*, pages 246 257, 2004.

[21] W. Li, J. Han, and J. Pei. Cmar: Accurate and efficient classification based on multiple class-association rules. In *Proc. of the 1st IEEE International Conference on Data Mining* (*ICDM'01*),

pages 369 376, 2001.

[22] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. of the 4th International Conference on Knowledge Discovery and Data Mining* (*KDD*-98), pages 80 86, 1998.

[23] S. Morishita and J. Sese. Traversing lattice itemset with statistical metric pruning. In *Proc. of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (*PODS* 2000), pages 226 236, 2000.

[24] P. C. Nguyen, K. Ohara, H. Motoda, and T. Washio. Cl-GBI: A novel approachfor extracting typical patterns from graph-structured data. In *Proc. of the 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining* (*PAKDD* 2005), pages 639 649, 2005.

[25] K. Ohara, P. C. Nguyen, A. Mogi, H. Motoda, and T. Washio. Constructing decision trees based on chunkingless graph-based induction. In L. B. Holder andD. J. Cook, editors, *Mining Graph Data*, pages 203-226. Wiley-Interscience,2006.

[26] J. Park, M. Chen, and P. Yu. An effective hash-based algorithm for mining association rules. In *Proc. of the* 1995 *ACMSIGMOD International Conference on Management of Data*, pages 175 186, 1995.

[27] J. Pei, J. Han, and R. Mao. Closet: An efficient algorithm for mining frequent closed itemsets. In *Proc. of the* 2000 *ACM-SIGMOD International Workshop on Data Mining and Knowledge Discovery*, pages 11 20, 2000.

[28] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. C. Hsu. PrefixSpan: Mining sequential patterns efficiently by prefix projected pattern growth. In *Proc. of the 17th International Conference on Data Engineering* (*ICDE* 2001), pages 215 224, 2001.

[29] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. of the 21th International Conference on Very Large Data Bases* (*VLDB* 1995), pages 432 444. Morgan Kaufmann, 1995.

[30] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proc. of the 21th International Conference on Very Large Data Bases* (*VLDB* 1995),pages 407 419, 1995.

[31] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. of the* 1996 *ACM SIGMOD International Conference on Management of Data*, pages 1-12, 1996.

[32] R. Srikant and R. Agrawal. Mining sequential patterns: Generalization and performance improvement. In *Proc. of the 5th International Conference on Extending Database Technology*, pages 3-17, 1996.

[33] A. Termier, M. C. Rousset, and M. Sebag. Dryade: A new approach for discovering closed frequent trees in heterogeneous tree databases. In *Proc. of the 4th IEEE International Conference on Data Mining* (*ICDM*'04), pages 543-546,2004.

[34] T. Uno, T. Asai, Y. Uchida, and H. Arimura. An efficient algorithm for enumerating frequent closed patterns in transaction databases. In *Proc. of the 7th International Conference on Discovery Science* (*DS*'04), *LNAI* 3245, Springer,pages 16 30, 2004.

[35] T. Washio, H. Matsuura, and H. Motoda. Mining association rules for estimation and prediction. In *Proc. of the 2nd Pacific Asia Conference on Knowledge Discovery and Data Mining* (*PAKDD* 1998), pages 417 419, 1998.

[36] T. Washio, Y. Mitsunaga, and H. Motoda. Mining quantitative frequent itemsets using adaptive density-based subspace clustering. In *Proc. of the 5th IEEE International Conference on Data*

Mining (*ICDM*'05), pages 793 796, 2005.

[37] T. Washio, K. Nakanishi, and H. Motoda. Deriving class association rules basedon levelwise subspace clustering. In *Proc. of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases* (*PKDD* 2005), *LNAI* 3721, Springer, pages 692 700, 2005.

[38] G. Webb. Efficient search for association rules. In *Proc. of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99 107, 2000.

[39] G. Webb and S. Zhang. K-optimal rule discovery. *Data Mining and Knowledge Discovery*, Vol. 10, No. 1, pages 39 79, 2005.

[40] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*, *2nd Edition*. Morgan Kaufmann, San Francisco, 2005. http://www.cs.waikato.ac.nz/ml/weka/.

[41] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *Proc. of the 2nd IEEE International Conference on Data Mining* (*ICDM*'02), pages721 724, 2002.

[42] K. Yoshida and H. Motoda. Clip: Concept learning from inference pattern. *Journal of Artificial Intelligence*, Vol. 75, No. 1, pages 63 92, 1995.

[43] M. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No. 3, pages 372 390, 2000.

[44] M. J. Zaki. Efficiently mining frequent trees in a forest. In *Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD*'02), pages 71-80, 2002.

第 5 章　EM

Geoffrey J. McLachlan, Shu-Kay Ng

5.1　引言
5.2　算法描述
5.3　软件实现
5.4　示例
　　5.4.1　例 5.1：多元正态混合
　　5.4.2　例 5.2：混合因子分析
5.5　高级主题
5.6　习题
参考文献

摘要

期望最大化(EM)算法是一种被广泛用于极大似然(ML)估计的迭代型计算方法，它对处理大量的数据不完整问题非常有用。特别是，EM 算法能大幅度简化对有限混合模型 ML 拟合问题的处理，而混合模型是对聚类分析、模式识别等任务中的异质性进行建模的重要手段。EM 算法具有很多非常诱人的性质，包括数值计算的稳定性、实现上的简单性、可靠的全局收敛性等。为了处理数据挖掘应用中遇到的各种复杂问题，有必要对 EM 算法进行适当的扩展，但精髓仍然是要保持其简单性和稳定性。

5.1 引 言

近年来，在数据挖掘、机器学习和模式识别等领域的算法研究中，EM 算法吸引了很多人的兴趣[20][27][28]。最初由 Dempster 等人发表的关于 EM 算法的论文[8]极大激发了人们使用有限混合分布对异质性数据进行建模的兴趣。混合模型的极大似然(ML)拟合是一个经典问题，EM 算法将其抽象为更一般性的不完整数据 ML 估计问题[20]，从而大大简化了该问题的求解。极大似然值估计和基于似然度的推断在统计理论和数据分析任务里居于中心地位。极大似然值估计方法非常通用而且有很多吸引人的特质[6, 13, 31]。有限混合分布是一种非常灵活的数学方法，可以用于对随机现象生成的数据进行建模和聚类。本章我们将重点放在 ML 框架下如何使用 EM 算法估计有限混合模型。

为了阐明基于混合模型的聚类方法，我们假设有观测数据 $y_1,\cdots,y_n$，其维数是 p，这些观测数据来自一个含有 g 个(已知参数)分量的混合分布，该混合分布的分量的权重(未知参数)记为 $\pi_1,\cdots,\pi_g$，其和为 1。则观测数据 y_j 的混合密度可以表示为：

$$f(y_j;\Psi)=\sum_{i=1}^{g}\pi_i f_i(y_j;\theta_i)\quad (j=1,\cdots,n) \tag{5.1}$$

其中分量密度 $f_i(y_j;\theta_i)$ 由参数向量 θ_i(未知参数)确定，这样，全部未知参数的向量就是：

$$\Psi=(\pi_1,\cdots,\pi_{g-1},\theta_1^T,\cdots,\theta_g^T)^T$$

其中上标 T 表示向量转置。

那么有限混合模型的估计就归结为对参数向量 Ψ 的估计，通常可以借助于 ML 估计方法。数学上这是一个最优化问题，其优化的目标函数是似然度 $L(\Psi)$或者等价对数似然度 $\log L(\Psi)$，其定义域是整个参数取值空间。根据最优化原理，参数 Ψ 的 ML 估计$\hat{\Psi}$是对数似然度一阶导方程的根：

$$\partial \log L(\Psi)/\partial\Psi=0 \tag{5.2}$$

其中 $\log L(\Psi)=\sum_{j=1}^{n}\log f(y_j;\Psi)$ 是对数似然函数，这里需要假设 $y_1,\cdots,y_n$ 是互相独立的。

对于每一个 n，ML 都能确定一个估计$\hat{\Psi}$[13]，因此就可以得到式(5.2)的一致、渐进和有效的根序列。只需满足一些适当的规则条件，即可确保该序列的存在[7]。这些根以趋进于 1 的概率对应于参数空间内的局部极大值点。对于模型估计问题而言，似然度在参数空间内通常会存在一个全局极大值点。如果让估计$\hat{\Psi}$在每一个 n 上都能使得 $L(\Psi)$取得全局

最大值，那么式(5.2)的根序列就会具有良好的渐进收敛性，在这种情况下，$\hat{\Psi}$就是一个极大似然估计 MLE[18]。在后面我们提到的估计$\hat{\Psi}$都是 MLE，即使它没有得到似然度的全局的极大值。事实上，即便在 5.4.1 节混合模型的例子中似然度无界的情形下，只要满足一些常见的规则条件，式(5.2)的一致、有效和渐进正态的根序列就仍能存在[16]。

5.2 算法描述

文献[18]简述了 EM 算法的历史，该算法是一种处理数据不完整问题的迭代型算法，每次迭代包含两个步骤：期望步骤(E-Step)和最大化步骤(M-Step)。这里，用 $y=(y_1^T,\cdots,y_n^T)^T$ 表示观测数据向量，用 z 表示缺失数据向量，用 $x=(y^T,z^T)^T$ 表示完整数据向量。EM 算法通过对“含完整数据”的对数似然函数值 $\log L_c(\Psi)$的逐步迭代计算来求解“含不完整数据”的式(5.2)。由于 $\log L_c(\Psi)$依赖于不可观测的缺失数据 z，所以在 E-Step 中将 $\log L_c(\Psi)$用所谓的 Q 函数来代替，也就是基于当前 Ψ 值以 y 为条件差别大。更确切地说，在 EM 算法的第 k+1 次迭代中，E-Step 执行的计算是：

$$Q(\Psi;\Psi^{(k)}) = E_{\Psi^{(k)}}\{\log L_c(\Psi) \mid y\}$$

其中 $E_{\Psi^{(k)}}$表示使用参数向量 $\Psi^{(k)}$的期望。M-Step 的任务是更新 Ψ 的估计值 $\Psi^{(k+1)}$，从而使得在 Ψ 的整个参数空间上 $Q(\Psi;\Psi^{(k)})$函数取最大值[18]。如此，E-Step 和 M-Step 交替循环执行直到对数似然度的变化小于某些预定的阈值。就像 5.1 节提到的，EM 算法的每一轮迭代都会稳定地增加似然度的值：

$$L(\Psi^{(k+1)}) \geqslant L(\Psi^{(k)})$$

如果“含完整数据”的概率密度函数是指数族分布的话，那么 E-Step 和 M-Step 的形式就会很简单[18]。在很多实际问题中，M-Step 存在封闭形式的解。但是有些情况下很难给出解的封闭形式，这就难以找到使 $Q(\Psi;\Psi^{(k)})$函数全局最大的解。在这种情况下，人们提出一种称为广义 EM(GEM)的算法，该算法放松了 M-Step，仅要求 $\Psi^{(k+1)}$能增加 $Q(\Psi;\Psi^{(k)})$函数的值即可，即：

$$Q(\Psi^{(k+1)};\Psi^{(k)}) \geqslant Q(\Psi^{(k)};\Psi^{(k)})$$

EM 算法存在一些不足：(1)它不能自动生成参数估计值的协方差矩阵。不过不要紧，只需结合一些合适的方法就可以避免这个问题[18]；(2)有些情况下算法收敛非常缓慢；(3)在有些问题中，E-Step 或 M-Step 是不可解析的。我们将在 5.5 节中对最后两个点进行简要讨论。

5.3 软件实现

EMMIX 程序：McLachlan 等人[22]针对混合多变量正态分布或混合多分量 t-分布的 ML 拟合问题开发了一个基于 EM 算法的通用工具 EMMIX。该软件用于处理连续型多变量数据，包括许多对混合模型拟合非常必要的支持，如为 EM 算法提供初始值；通过多种方法为混合模型的参数拟合提供标准误；具体见下文。

初始值：当对数似然度有多个对应局部极大值的根时，EM 算法应从大范围多次选择

初始值以搜索所有可能的局部极大值。对于有限混合模型的情况，初始参数值的获取可使用 k-means 聚类算法、层次聚类方法或随机的数据分区等方法[20]。EMMIX 程序提供了一个随机初始值的选项，每一次选初始值，用户先对数据进行欠采样，然后据此确定随机初始值。这样做是为了限制中心极限定理的效应，否则大样本中的每个分量的随机选择的初始值都相近[20]。

标准误：在 EM 算法的一些文献中[11,15,18]，提出了多种增强 EM 的方法，要点是计算 ML 估计的协方差矩阵。另外，还可以使用 EMMIX 程序的自举重采样（参数化或非参数化版本）来计算标准误[18,20]。

分量数：我们可以通过考察似然函数来选择一个适当的分量（聚簇）数 g。当没有任何关于数据中的聚簇数目的先验信息时，我们可以在增大 g 的过程中监测对数似然函数的增大情况。在任何阶段，都可依据似然比检验或基于信息论的判别准则，如贝叶斯信息准则（BIC），来决定选择 $g=g_0$ 还是 $g=g_0+1$。但不幸的是，对于似然比检验方法的统计量 λ，不能保证其卡方分布的零假设的自由度等于两个分别有 $g=g_0+1$ 和 $g=g_0$ 个分量的混合模型的参数数量的差值 d。EMMIX 程序提供了一种自举重采样方法来评估零（假设）分布（也就是 p-值）的统计量（$-2\log\lambda$）。另外，虽然不能从理论上保证 BIC 的有效性，我们还是会使用这一准则，即如果 $-2\log\lambda$ 大于 $d\log(n)$，就选择 $g=g_0+1$，否则就选 $g=g_0$。

其他混合建模软件：还有一些用 EM 算法求解的基于 ML 估计的混合建模软件。例如，Fraley 和 Raftery[9]开发的 MCLUST 程序，该工具通过对分量-协方差矩阵的进行多种形式的参数化，进而实现了基于正态分量的层次聚类。该工具被移植到商业软件 S-PLUS 中，并且还有建模背景（泊松）噪声的可选功能。读者可以在 McLachlan 和 Peel[20]提供的参考文献附录中找到很多拟合混合模型的软件。

5.4　示　　例

在本节中，我们举两个例子来演示在一些常见的数据挖掘的情形中，EM 算法是如何得到的 ML 估计的。这两个例子都关于 EM 算法处理有限混合模型的 ML 估计问题的，我们知道有限混合模型在异质性数据建模中很常用。这两个例子将演示在“不完整数据”形式化框架下，EM 算法如何获得 ML 估计。

5.4.1　例 5.1：多元正态混合

这个例子讲的是用 EM 算法对有限混合（分量为多元正态分布）模型进行 ML 估计[20]。参见式(5.1)，y_j 的混合密度为：

$$f(y_j;\Psi)=\sum_{i=1}^{g}\pi_i\phi(y_j;\mu_i,\Sigma_i)\quad(j=1,\cdots,n)\tag{5.3}$$

其中 $\phi(y_j;\mu_i,\Sigma_i)$ 表示 p 维多元正态分布，μ_i 是均值，Σ_i 是协方差矩阵。未知参数向量 Ψ 由 3 部分构成：混合比例 $\pi_1,\cdots,\pi_{g-1}$；每个分量的均值 μ_i；每个分量的（分量）协方差矩阵 Σ_i。基于 Ψ 的对数似然度为：

$$\log L(\Psi) = \sum_{j=1}^{n} \log\Big\{ \sum_{i=1}^{g} \pi_i \phi(y_j; \mu_i, \Sigma_i) \Big\}$$

用 EM 算法的迭代过程可以得到对数似然度的局部极大值。

在 EM 框架下，每个 y_j 被认为来自混合模型(式(5.3))的 g 个分量中一个。用 $z_1, \cdots, z_n$ 表示不可观测的分量指示向量，其中 z_j 的第 i 个元素 z_{ij} 的值取 1 还是 0，取决于第 j 个观测量 y_j 是否来自第 i 个分量。注意，可观测的数据向量 y 不是完整的数据，因为还有相关但未知的分量指示向量 $z_1, \cdots, z_n$ 数据。所以，含完整数据的向量为 $x=(y^T, z^T)^T$，其中 $z=(z_1^T, \cdots, z_n^T)^T$，基于参数 Ψ 的完整数据对数似然度为：

$$\log L_c(\Psi) = \sum_{i=1}^{g} \sum_{j=1}^{n} z_{ij} \{\log \pi_i + \log \phi(y_j; \mu_i, \Sigma_i)\} \tag{5.4}$$

在 EM 算法运用到式(5.4)时，将 z_{ij} 当作缺失的数据。在第 k+1 次迭代时，E-Step 计算 Q 函数 $Q(\Psi; \Psi^{(k)})$，这是在给定 y 和当前 $\Psi^{(k)}$ 时完整数据对数似然度的条件期望。再记对应 z_{ij} 的随机变量为 Z_{ij}，由于完整数据对数似然度(式(5.4))关于缺失数据 z_{ij} 是线性的，所以，我们只需借助可观测数据 y，就能简单地计算出随机变量 Z_{ij} 当前的条件期望，即：

$$\begin{aligned} E_{\Psi^{(k)}}(Z_{ij} \mid y) &= \mathrm{pr}_{\Psi^{(k)}}\{Z_{ij} = 1 \mid y\} = \tau_i(y_j; \Psi^{(k)}) \\ &= \pi_i^{(k)} \phi(y_j; \mu_i^{(k)}, \Sigma_i^{(k)}) \Big/ \sum_{h=1}^{g} \pi_h^{(k)} \phi(y_j; \mu_h^{(k)}, \Sigma_h^{(k)}) \end{aligned} \tag{5.5}$$

其中 $i=1, \cdots, g; j=1, \cdots, n$。$\tau_i(y_j; \Psi^{(k)})$ 是第 j 个可观测数据 y_j 属于有限混合分布的第 i 个分量的后验概率。从式(5.4)和式(5.5)，我们可以得到：

$$Q(\Psi; \Psi^{(k)}) = \sum_{i=1}^{g} \sum_{j=1}^{n} \tau_i(y_j; \Psi^{(k)}) \{\log \pi_i + \log \phi(y_j; \mu_i, \Sigma_i)\} \tag{5.6}$$

对于正态密度分量型的混合，可以先行计算下面的几个充分统计量，这样对于后面的处理十分方便：

$$\begin{aligned} T_{i1}^{(k)} &= \sum_{j=1}^{n} \tau_i(y_j; \Psi^{(k)}) \\ T_{i2}^{(k)} &= \sum_{j=1}^{n} \tau_i(y_j; \Psi^{(k)}) y_j \\ T_{i3}^{(k)} &= \sum_{j=1}^{n} \tau_i(y_j; \Psi^{(k)}) y_j y_j^T \end{aligned} \tag{5.7}$$

对于正态分量，M-Step 存在封闭形式的解，而且可以用式(5.7)对其进行化简[20,26]，如下：

$$\begin{aligned} \pi_i^{(k+1)} &= T_{i1}^{(k)} / n \\ \mu_i^{(k+1)} &= T_{i2}^{(k)} / T_{i1}^{(k)} \\ \sum_i^{(k+1)} &= \{T_{i3}^{(k)} - T_{i1}^{(k)^{-1}} T_{i2}^{(k)} T_{i2}^{(k)^T}\} / T_{i1}^{(k)} \end{aligned} \tag{5.8}$$

如果不对分量协方差矩阵 Σ_i 进行限制，每个数据点都在参数空间的边缘产生奇异值，这将导致 $L(\Psi)$ 不再是有界的[16,20]。有一个问题需要引起注意，如果一个分量的广义方差(即协方差矩阵的行列式)非常小，找到的(伪)局部极大值往往会比较大。这种分量对应

的聚簇中会有一些聚合得很紧密的数据点，如果这些数据是多变量的，那它们几乎是处于一个低维子空间中。

实际上，通常将所有的分量协方差矩阵 Σ_i 都设置为同一个 $\Sigma_i = \sum (i = 1, \cdots, g)$，此处 Σ 具体值是未定的。如果混合分量又是正态分布的话，那么估计这个同质分量协方差矩阵的更新方法就是：

$$\sum^{(k+1)} = \sum_{i=1}^{g} T_{il}^{(k)} \Sigma_i^{(k+1)} \Big/ n \tag{5.9}$$

其中 $\Sigma_i^{(k+1)}$ 由式(5.8)给出。π_i 和 μ_i 的更新方法仍保持式(5.8)不变，即和异质分量协方差矩阵情况下计算方法一样。

我们在这里选用 UCI 机器学习资料库提供的著名的 Iris 数据集[1]。它包含了 setosa、versicolor 和 virginica 这 3 个品种(类别)的 50 个植物的度量数据，即“萼片长度，萼片宽度，花瓣长度，花瓣宽度”。在这里，我们不考虑数据类别(品种)而仅基于上述 4 维对这些数据进行聚类，使用 EMMIX 程序拟合含有 g=3 个正态分量的混合模型，它的分量协方差矩阵是异质对角矩阵[22]。这样的话，未知参数向量 Ψ 就包括：(1)混合比例参数 π_1，π_2；(2)分量均值 μ_i；(3) 分量协方差矩阵对角线元素 $\Sigma_i (i = 1,2,3)$。参数向量 Ψ 的初始值 $\Psi^{(0)}$ 设为以下值：

$$\pi_1^{(0)} = 0.31 \quad \pi_2^{(0)} = 0.33, \quad \pi_3^{(0)} = 0.36$$

$$\mu_1^{(0)} = (5.0, 3.4, 1.5, 0.2)^T,$$

$$\mu_2^{(0)} = (5.8, 2.7, 4.2, 1.3)^T$$

$$\mu_3^{(0)} = (6.6, 3.0, 5.5, 2.0)^T$$

$$\Sigma_1^{(0)} = \mathrm{diag}(0.1, 0.1, 0.03, 0.01)$$

$$\Sigma_2^{(0)} = \mathrm{diag}(0.2, 0.1, 0.2, 0.03)$$

$$\Sigma_3^{(0)} = \mathrm{diag}(0.3, 0.1, 0.3, 0.1)$$

这些值是使用 k-means 聚类方法得到的。EMMIX 程序的默认停止准则是：(本次迭代的对数似然度-向前第 10 次迭代的对数似然度)/本次迭代的对数似然度<0.000 001。本例子的 EM 算法运行结果见表 5.1。Ψ 的最大似然估计 $\Psi^{(k)}$ 在 EM 迭代次数达到 k=29 就可用了。也可以不用 k-means 设置初始值，EMMIX 程序有自己的 EM 算法初始值自动设置功能。比如，采用 10 次随机启动(对数据进行 70%的欠采样)，10 次 k-means 启动，6 次层次法启动来确定初始值 $\Psi^{(0)}$，参见 5.3 节和文献[22]。不论哪种初始化方法，Ψ 的最后估计都与表 5.1 中的结果相同。

表 5.1　例 5.1 的 EM 算法的结果

迭代	$\pi_i^{(k)}$	$\mu_i^{(k)T}$	$\sum_i^{(k)}$ 的对角线元素	对数似然度
0	0.310	(5.00, 3.40, 1.50, 0.20)	(0.100, 0.100, 0.030, 0.010)	−317.984 21
	0.330	(5.80, 2.70, 4.20, 1.30)	(0.200, 0.100, 0.200, 0.030)	
	0.360	(6.60, 3.00, 5.50, 2.00)	(0.300, 0.100, 0.300, 0.100)	

续表

迭代	$\pi_i^{(k)}$	$\mu_i^{(k)T}$	$\Sigma_i^{(k)}$ 的对角线元素	对数似然度
1	0.333	(5.01,3.43,1.46,0.25)	(0.122,0.141,0.030,0.011)	−306.909 35
	0.299	(5.82,2.70,4.20,1.30)	(0.225,0.089,0.212,0.034)	
	0.368	(6.62,3.01,5.48,1.98)	(0.322,0.083,0.325,0.088)	
2	0.333	(5.01,3.43,1.46,0.25)	(0.122,0.141,0.030,0.011)	−306.873 70
	0.300	(5.83,2.70,4.21,1.30)	(0.226,0.087,0.218,0.034)	
	0.367	(6.62,3.01,5.47,1.98)	(0.323,0.083,0.328,0.087)	
10	0.333	(5.01,3.43,1.46,0.25)	(0.122,0.141,0.030,0.011)	−306.862 34
	0.303	(5.83,2.70,4.22,1.30)	(0.227,0.087,0.224,0.035)	
	0.364	(6.62,3.02,5.48,1.99)	(0.324,0.083,0.328,0.086)	
20	0.333	(5.01,3.43,1.46,0.25)	(0.122,0.141,0.030,0.011)	−306.860 75
	0.304	(5.83,2.70,4.22,1.30)	(0.228,0.087,0.225,0.035)	
	0.363	(6.62,3.02,5.48,1.99)	(0.324,0.083,0.327,0.086)	
29	0.333	(5.01,3.43,1.46,0.25)	(0.122,0.141,0.030,0.011)	−306.860 52
	0.305	(5.83,2.70,4.22,1.30)	(0.229,0.087,0.225,0.035)	
	0.362	(6.62,3.02,5.48,1.99)	(0.324,0.083,0.327,0.085)	

5.4.2 例 5.2：混合因子分析

McLachlan 和 Peel[21]采用混合因子分析模型对 wine 数据集进行聚类，该数据集来自 UCI 机器学习资料库。该数据集给出了生长在意大利同一地区的三个品种葡萄酒的 p=13 种化学成分，共有 n=178 条数据。此处用含有 3 个正态分量的混合模型在对该数据集进行聚类。但是，我们看到，算起来每个(不受限) 分量协方差矩阵 Σ_i(i = 1,2,3) 有 91 参数(下三角矩阵的面积 13 × 14/2 = 91)，整个混合模型的参数总量(大于 3×91)相比样本规模 N=178 显得太大了。混合因子分析法能有效减少需要拟合的参数数量。在混合因子分析法中，将每个观测对象 Y_j(j=1,…,n)建模为：

$$Y_j = \mu_i + B_i U_{ij} + \in_{ij}$$

其中，U_{ij}是一个 q(q<p)维的隐含/不可观测的因子向量，B_i 是一个 p×q 维的因子负载矩阵(模型参数)。因子 U_i,…,U_{in}是独立的标准正态分布 N(0,I_q)，其中 I_q 是 q * q 的单位矩阵；$\in_{ij}$是独立的正态分布 N(0,D_i)，D_i 是一个 p * p 的对角矩阵(i=1,…,g)；U_{ij}和 $\in_{ij}$是相互独立的。这样就有：

$$f(y_j;\Psi) = \sum_{i=1}^{g} \pi_i \phi(y_j;\mu_i,\Sigma_i)$$

其中

$$\Sigma_i = B_i B_i^T + D_i \quad (i = 1,\cdots,g)$$

未知参数向量现在的构成就变为：U_i、B_i 和 D_i，以及分量混合比例 $\pi_i(i=1,\cdots,g-1)$。

交替期望条件最大化(AECM)算法[24]适用于混合因子分析模型的 ML 估计(请参见 5.5 节)。可将未知参数分为 $(\Psi_1^T,\Psi_2^T)^T$ 两部分，Ψ_1 包含 $\pi_i(i=1,\cdots,g-1)$ 和 $\mu_i(i=1,\cdots,g)$，而 Ψ_2 包含 B_i 和 $D_i(i=1,\cdots,g)$。AECM 算法是期望条件最大化(ECM)算法[23]的扩展，该算法在不同条件最大化(CM)步骤上"完整数据"的指定可以不同。在本例中，对应上面的参数划分，每次迭代包含两轮，每轮包含一次 E-Step 和一次 CM-Step。在 AECM 算法的第一轮里，我们仅将成分指示向量 $z_1,\cdots,z_n$ 指定为缺失数据，参见式(5.4)。在第 $k+1$ 次迭代上，第一轮中的 E-Step 本质上就是式(5.5)和式(5.6)；第一轮中的 CM-Step 要更新 $\Psi_1^{(k+1)}$：

$$\pi_i^{(k+1)} = \sum_{j=1}^{n}\tau_{ij}^{(k)}\Big/n$$

$$\mu_i^{(k+1)} = \sum_{j=1}^{n}\tau_{ij}^{(k)}y_j\Big/\sum_{j=1}^{n}\tau_{ij}^{(k)}$$

在第二轮里，我们将因子 $U_{i1},\cdots,U_{in}$ 和成分指标向量 $z_1,\cdots,z_n$ 同时指定为缺失数据。令 $\Psi^{(k+1/2)}$ 等于 $(\Psi_1^{(k+1)^T},\Psi_2^{(k)^T})^T$，在第二轮里的 E-Step 计算条件期望值的公式为：

$$E_{\Psi^{(k+1/2)}}\{Z_{ij}(U_{ij}-\mu_i)\mid y_j\} = \tau_{ij}^{(k+1/2)}\gamma_i^{(k)^T}(y_j-\mu_i)$$

$$E_{\Psi^{(k+1/2)}}\{Z_{ij}(U_{ij}-\mu_i)(U_{ij}-\mu_i)^T\mid y_j\}$$

$$=\tau_{ij}^{(k+1/2)}\{\gamma_i^{(k)^T}(y_j-\mu_i)(y_j-\mu_i)^T\gamma_i^{(k)}+\Omega_i^{(k)}\}$$

其中

$$\gamma_i^{(k)} = (B_i^{(k)}B_i^{(k)^T}+D_i^{(k)})^{-1}B_i^{(k)}$$

$$\Omega_i^{(k)} = I_q-\gamma_i^{(k)^T}B_i^{(k)}$$

这个 E-Step 用到了一个条件分布，即给定 y_j 和 $Z_{ij}=1$ 条件时 U_{ij} 的分布，如下：

$$U_{ij}\mid y_j,z_{ij}=1\sim N(\gamma_i^T(y_j-\mu_i),\Omega_i)$$

在第二轮里的 CM-Step 对 $\Psi_2^{(k+1)}$ 的估计公式为：

$$B_i^{(k+1)} = V_i^{(k+1/2)}\gamma_i^{(k)}(\gamma_i^{(k)^T}V_i^{(k+1/2)}\gamma_i^{(k)}+\Omega_i^{(k)})^{-1}$$

$$D_i^{(k+1)} = \mathrm{diag}\{V_i^{(k+1/2)}-B_i^{(k+1)}H_i^{(k+1/2)}B_i^{(k+1)^T}\}$$

其中

$$V_i^{(k+1/2)} = \frac{\sum_{j=1}^{n}\tau_{ij}^{(k+1/2)}(y_j-\mu_i^{(k+1)})(y_j-\mu_i^{(k+1)})^T}{\sum_{j=1}^{n}\tau_{ij}^{(k+1/2)}}$$

$$H_i^{(k+1/2)} = \gamma_i^{(k)^T}V_i^{(k+1/2)}\gamma_i^{(k)}+\Omega_i^{(k)}$$

为了能说明问题，本例中选取不同 q 值对 wine 数据集进行了混合因子分析模型的拟合，实验中没有用到类别信息。为了对 Ψ 进行初始化，本例中 EMMIX 程序使用 10 次随机初始化(对数据集进行 70%的欠采样)，对不受限分量协方差矩阵的正态混合模型进行拟合。首先，可以获得对 π_i 和 μ_i 的估计，AECM 算法用此估计作为 π_i 和 μ_i 初始值。再有，可以获得对 Σ_i 的估计(记为 Σ_i^0)，就取其对角元素生成对角矩阵 D_i 的初始值 $D_i^{(0)}$。最后，B_i

的初始值可以用文献[20]中描述的方法获得。表 5.2 展示了 AECM 算法取 q=1 一直到 q=8 的运行结果。并且,我们还给出了似然比检验统计量 $-2\log\lambda$:用拟合出的含有 q 个因子到含有 q+1 个因子的两个混合模型的对数似然度的增量再乘以 2。给定分量的数目 g,零假设统计量 $-2\log\lambda$ 的渐进分布就服从自由度为 d 的卡方分布,其中 d 是零假设分布参数数量(对应 p)与备择假设分布参数数量(对应 q)之差。从表 5.2 可以看到,在 q=2 或 3 时聚类的错误率是最小的,但是在聚类任务中错误率其实是不知道的,因此不能用来指导 q 的选择。可用似然比检验来确定因子数量 q,方法是把 $-2\log\lambda$ 输入有 $d=g(p-q_0)=21$ 个自由度的卡方分布,如选 $q=q_0=6$ 作为零假设,检验得到的结果 P=0.28,表明 $q=q_0=6$ 与 $q=q_0+1=7$ 相比并不显著。

表 5.2 例 5.2 的 AECM 算法的结果

q	对数似然度	错误(%错误率)	$-2\log\lambda$
1	−3102.254	2(1.12)	—
2	−2995.334	1(0.56)	213.8
3	−2913.122	1(0.56)	164.4
4	−2871.655	3(1.69)	82.93
5	−2831.860	4(2.25)	79.59
6	−2811.290	4(2.25)	41.14
7	−2799.204	4(2.25)	24.17
8	−2788.542	4(2.25)	21.32

5.5 高级主题

在本节中,我们将对 EM 算法进行一些扩展,使其可以处理一些更困难的问题(涉及 E-Step 或 M-Step 的计算),以及加快算法收敛速度。此外,我们还简要介绍一下 EM 算法在隐马尔可夫模型(HMM 模型)上的应用,该方法提供了一种将混合模型推广到非独立数据的简单途径。

在 EM 算法用于广义线性混合模型等场合时,E-Step 计算很复杂而且不能确保 Q 函数存在封闭形式的解。这种情况下可以用蒙特卡洛(MC)过程计算 E-Step。在进行(k+1)次迭代时,E-Step 涉及:

(1) 用条件分布 $g(z|y;\Psi^{(k)})$ 模拟出缺失数据 Z 的 M 份独立数据集;

(2) 近似 Q 函数

$$Q(\Psi;\Psi^{(k)})\approx Q_M(\Psi;\Psi^{(k)})=\frac{1}{M}\sum_{m=1}^{M}\log L_c(\Psi;y,z^{(m_k)})$$

其中,$z^{(m_k)}$ 是基于 $\Psi^{(k)}$ 的第 m 个缺失值。在 M-Step 中,计算使得 Q 函数取最大值的 $\Psi^{(k+1)}$。称以上这种 EM 变体为蒙特卡洛期望最大化(MCEM)算法[33]。因为该算法会引

入 MC 错误到 E-Step 中，所以导致其单调性丧失。但是通常情况下，该算法能以很高的概率趋近极大值点[4]。该算法中 M 值的确定以及收敛判定对于实际使用来说是至关重要的，参见[4,18,33]。

EM 算法的 M-Step 仅涉及完整数据的 ML 估计，所以计算通常比较简单。但是在混合因子分析(5.4.2 节)等应用中对应的 M-Step 是相当复杂的。此时，如果能对被估计参数的某个函数进行计算，就可以简化 M-Step，ECM 算法[23]就是基于这个思路对 EM 算法进行了扩展。ECM 算法将 EM 算法的一个复杂 M-Step 替换成几个简单的 CM-Step，保持了完整数据条件最大化计算的简单性。值得一提的是，ECM 算法还保留了 EM 算法的收敛性[18,23]。在 5.4.2 节中提到的 AECM 算法[24]在此基础上又进一步，允许在一次迭代内或迭代间的不同 CM-Step 上使用的"完整数据"的定义不一样。这种灵活的数据增强和模型约减的框架特别适合参数数量大的混合因子分析一类的应用。

现在以百万计的大规模多维数据集是很常见的，所以加快 EM 算法的收敛速度也成为大数据处理的一个迫切需求。但同时要注意，只有能保留其简单性和稳定性才有意义。Neal 和 Hinton[25] 提出了一个增量式 EM(IEM)算法加快了 EM 算法的收敛速度。该算法先将数据划分成 B 个块，然后每次只对一个块执行(部分)E-Step 和对全部数据执行一个(完整)M-Step。也就是说，IEM 算法一次"扫描"包括 B 个"部分 E-Step"和 B 个"完整 M-Step"[26]。在 5.6 节的习题 6 和 7 表明 IEM 算法收敛通常仅需较少的扫描遍数，因此一般速度比 EM 算法快。IEM 算法每一次扫描也是增加的似然度值，参见文献[27]的讨论。

在混合框架下，有观测数据 $y_1,\cdots,y_n$ 和通常称为"隐变量"的不可观测的分量指示向量 $z=(z_1^T,\cdots,z_n^T)^T$。在语音识别应用中，未知的 z_j 是串行依赖的典型频谱，而可观测的语音信号 $y_j(j=1,\cdots,n)$ 要依赖于它。因此，值序列(或值集)中的 z_j 不能被视为相互独立。在自动语音识别(AVR)或自然语言处理(NLP)中，通常使用一个基于有限状态空间的马尔可夫模型描述隐变量 z 的分布 [18]。由于 z 是一个隐含的依赖结构，y_j 的密度就不能再像式(5.1)那样被简单表示成独立分量混合的形式。但是，可以假设 $y_1,\cdots,y_n$ 在给定 $z_1,\cdots,z_n$ 条件下是独立的：

$$f(y_1,\cdots,y_n \mid z_1,\cdots,z_n;\theta)=\prod_{j=1}^{n} f(y_j \mid z_j;\theta)$$

其中每个条件分布是不同的，θ 表示全部条件分布的未知参数向量。在 HMM 的一些文献中，称此类问题的求解算法为 Baum-Welch 算法，最初由 Baum 和他的合作者提出并证明其收敛性，这些工作早于 Dempster 等人提出 EM 算法的时间[8]，参见文献[2]。Baum-Welch 算法的 E-Step 是确切型的，但它要求在数据上进行前向和后向的递归运算[18]，其 M-Step 存在封闭形式的解，就是将多项式分布参数的极大似然估计和马尔可夫链的转移概率进行组合，参见文献[14,30]。

5.6 习 题

本章所列 10 道关于 EM 算法及其变体习题取自多个数据挖掘和模式识别应用领域。需要提醒的是，这些习题中的数据不完整性可能不像前文的 5.4 示例中那样明显和自然。

(1) Böhning 等人[3]对泰国北部地区 1982 年至 1985 年间的 602 名学龄前儿童的健康

状况进行了研究[32]，表 5.3 显示了在其研究周期内的患病(发烧、咳嗽，或两者皆有)频次信息。

表 5.3 泰国北部地区学龄前儿童样本的患病频率

病人序号	频率	病人序号	频率	病人序号	频率
0	120	8	25	16	6
1	64	9	19	17	5
2	69	10	18	18	1
3	72	11	18	19	3
4	54	12	13	20	1
5	35	13	4	21	2
6	36	14	3	23	1
7	25	15	6	24	2

如果用含有 3 个泊松分量的混合分布来拟合该数据，就有如下的对数似然函数：

$$\log L(\Psi) = \sum_{j=1}^{n} \log\left\{ \sum_{i=1}^{3} \pi_i f(y_j, \theta_i) \right\}$$

此处 $\Psi=(\pi_1,\pi_2,\theta_1,\theta_2,\theta_3)^T$，$f(y_j,\theta_i)=\exp(-\theta_i)\theta_i^{y_i}/y_j(i=1,2,3)$，参照 5.4.1 节，记 y_j 属于第 i 个分量的后验概率为：

$$\tau_i(y_j;\Psi^{(k)}) = \pi_i^{(k)} f(y_j,\theta_i^{(k)}) \Big/ \sum_{h=1}^{3} \pi_h^{(k)} f(y_j,\theta_h^{(k)}) \quad (i=1,2,3)$$

证明 M-Step 中参数估计的更新公式为：

$$\pi_i^{(k+1)} = \sum_{j=1}^{n} \tau_i(y_j;\Psi^{(k)}) \Big/ n \quad (i=1,2)$$

$$\theta_i^{(k+1)} = \sum_{j=1}^{n} \tau_i(y_j;\Psi^{(k)}) y_j \Big/ (n\pi_i^{(k+1)}) \quad (i=1,2,3)$$

取参数的初始估计为：$\pi_1=0.6$，$\pi_2=0.3$，$\theta_1=2$，$\theta_2=9$，$\theta_3=17$，请计算 Ψ 的最大似然估计。

(2) 由 McLachlan 和 Peel [19]提出的(多元)t-分量混合分布的拟合也为正态分量混合模型提供了一种健壮的拟合方法。有 g 个 t 分量的混合分布的形式是：

$$f(y_j;\Psi) = \sum_{i=1}^{g} \pi_i f\left(y_j;\mu_i,\sum_i,v_i\right)$$

多元 t 分量密度 $f\left(y_j;\mu_i,\sum_i,v_i\right)$具有位置 μ_i、正定内积矩阵 $\sum_i$ 和自由度 andv_i(i = 1,…,g)[19, 29]。未知参数向量为：

$$\Psi = (\pi_1,\cdots,\pi_{g-1},\theta^T,v^T)^T$$

此处 $v=(v_1,\cdots,v_g)^T$ 是 t 分布的自由度，$\theta=(\theta_1^T,\cdots,\theta_g^T)^T$，$\theta_i$ 包括了 elements of μ_i 和 $\sum_i$ 的元素(i = 1,…,g)。参照 5.4.1 节，添加了分量指示向量 $z_1,\cdots,z_n$ 的观测数据仍是不完整的，再加上缺失数据 $u_1,\cdots,u_n$ 就构成完整的数据向量了，即：

$$x = (y^T, z_1^T, \cdots, z_n^T, u_1, \cdots, u_n)^T$$

$u_1, \cdots, u_n$使得对于给定的 $z_{ij}=1$ 时有 n 个(j=1,…,n)独立分布：

$$Y_j \mid u_j, z_{ij} = 1 \sim N\left(\mu_i, \sum_i / u_j\right)$$

和

$$U_j \mid z_{ij} = 1 \sim \text{gamma}\left(\frac{1}{2}v_i, \frac{1}{2}v_i\right)$$

证明完整数据的对数似然度包含 3 项：

$$\log L_c(\Psi) = \log L_{1c}(\pi) + \log L_{2c}(v) + \log L_{3c}(\theta) \tag{5.10}$$

其中

$$\log L_{1c}(\pi) = \sum_{i=1}^{g}\sum_{j=1}^{n} z_{ij}\log\pi_i$$

$$\log L_{2c}(v) = \sum_{i=1}^{g}\sum_{j=1}^{n} z_{ij}\left\{-\log\Gamma\left(\frac{1}{2}v_i\right)+\frac{1}{2}v_i\log\left(\frac{1}{2}v_i\right)+\frac{1}{2}v_i(\log u_j - u_j) - \log u_j\right\}$$

$$\log L_{3c}(\theta) = \sum_{i=1}^{g}\sum_{j=1}^{n} z_{ij}\left\{-\frac{1}{2}p\log(2\pi) - \frac{1}{2}\log\left|\sum_i\right| - \frac{1}{2}u_j\delta(y_j, \mu_i, ; \Sigma_i)\right\}$$

此处 $\delta\left(y_j, \mu_i; \sum_i\right) = (y_i - \mu_i)^T \sum_i^{-1} (y_j - \mu_i)$。

(3) 参见习题 2 中的混合 t 分布，证明 EM 算法在第(k+1)轮迭代的 E-Step 中，需要为所有的 i=1,…,g;j=1,…,n 进行如下的计算：

$$E_{\Psi^{(k)}}(Z_{ij} \mid y) = \tau_{ij}^{(k)} = \frac{\pi_i^{(k)} f\left(y_j; \mu_i^{(k)}, \sum_i^{(k)}, v_i^{(k)}\right)}{f(y_j; \Psi^{(k)})} \tag{5.11}$$

$$\sum\nolimits_{\Psi^{(k)}}(U_j \mid y, z_{ij} = 1) = u_{ij}^{(k)} = \frac{v_i^{(k)} + p}{v_i^{(k)} + \delta\left(y_j, \mu_i^{(k)}; \sum_i^{(k)}\right)} \tag{5.12}$$

$$E_{\Psi^{(k)}}(\log U_j \mid y, z_{ij} = 1) = \log u_{ij}^{(k)} + \left\{\psi\left(\frac{v_i^{(k)} + p}{2}\right) - \log\left(\frac{v_i^{(k)} + p}{2}\right)\right\} \tag{5.13}$$

此处 $\psi(r) = \{\partial\Gamma(r)/\partial r\}/\Gamma(r)$是 Digamma 函数[29]。提示：式(5.12)里，注意 gamma 分布是 U_j 的共轭先验分布；式(5.13)里，注意如果随机变量 S 服从 gamma(α,β)分布，就有 E(logS)=ψ(α)−logβ。

从式(5.10)可以在 M-Step 中独立地计算各个变量 $\pi^{(k+1)}$、$\theta^{(k+1)}$和 $v^{(k+1)}$。证明 $\pi^{(k+1)}$、$\theta^{(k+1)}$的更新公式如下：

$$\pi_i^{(k+1)} = \sum_{j=1}^{n}\tau_{ij}^{(k)} / n$$

$$\mu_i^{(k+1)} = \sum_{j=1}^{n}\tau_{ij}^{(k)} u_{ij}^{(k)} y_j \Big/ \sum_{j=1}^{n}\tau_{ij}^{(k)} u_{ij}^{(k)}$$

$$\sum_i^{(k+1)} = \frac{\sum_{j=1}^{n}\tau_{ij}^{(k)} u_{ij}^{(k)} (y_j - \mu_i^{(k+1)})(y_j - \mu_i^{(k+1)})^T}{\sum_{j=1}^{n}\tau_{ij}^{(k)}}$$

自由度 $v_i^{(k+1)}$ 的更新需要迭代处理，根据式(5.10) that $v_i^{(k+1)}$ 是如下方程的解：

$$\left\{-\psi\left(\frac{1}{2}v_i\right)+\log\left(\frac{1}{2}v_i\right)+1+\frac{1}{n_i^{(k)}}\sum_{j=1}^{n}\tau_{ij}^{(k)}(\log u_{ij}^{(k)}-u_{ij}^{(k)})\right.$$
$$\left.+\psi\left(\frac{v_i^{(k)}+p}{2}\right)-\log\left(\frac{v_i^{(k)}+p}{2}\right)\right\}=0$$

其中 $n_i^{(k)}=\sum_{j=1}^{n}\tau_{ij}^{(k)}(i=1,\cdots,g)$。

(4) 程序 EMMIX[22]有用于拟合多元 t 分量混合分布的选项。现在，要基于 Campbell 和 Mahon [5]的 Leptograpsus crab 数据集来拟合一个含有 2 个 t 分量（具有无限制标量矩阵 $\sum_i$ 和不相等自由度 v_i）的混合分布。在 crab 数据集里，将一个品种依据颜色再拆分成橙色和蓝色两个品种。每个品种的每个性别都有 50 个标本。这里关注的是对 orange crabs 的 100 个样本实例的 5 维度量(两个分量对应雄性和雌性)。以如下配置运行 EMMIX 程序：用取自 10 个随机启动点(使用 10%的欠采样)中的自动启动值，用 10 个聚簇的 k-means 启动点，用 6 层的层次化方法(在等标量矩阵和等自由度情况下，用户指定初始值 $v_1^{(0)}=v_2^{(0)}=13.193$)。请验证：v 的估计是 $\hat{v}_1=12.2$，$\hat{v}_2=300.0$，赋给每个分量的数量分别是 47 和 53(错分率 3%)。

(5) g 个广义线性模型(GLMs) 以比例 $\pi_1,\cdots,\pi_g$ 混合，第 j 个应变量 Y_j 的密度是：

$$f(y_j;\Psi)=\sum_{i=1}^{g}\pi_i f(y_j;\theta_{ij},k_i)$$

第 i 个分量的对数密度为

$$\log f(y_j;\theta_{ij},k_i)=k_i^{-1}\{\theta_{ij}y_j-b(\theta_{ij})\}+c(y_j;k_i)\quad(i=1,\cdots,g)$$

θ_{ij}是自然或典则参数，k_i 是分散参数。对第 i 个 GLM 分量，用 μ_{ij}表示 Y_j 的条件均值，用 $\eta_{ij}=h_i(\mu_{ij})=\beta_i^T x_j$ 表示线性预测器，$h_i(\cdot)$是链接函数，x_j 是第 j 个应变量 y_j 的解释变量向量[20]。用 $\Psi=(\pi_1,\cdots,\pi_{g-1},k_1,\cdots,k_g,\beta_1^T,\cdots,\beta_g^T)^T$ 表示未知参数向量，用 z_{ij}表示 5.4.1 节中定义的分量指示变量。E-Step 本质上就是式(5.5)和式(5.6)，不同之处仅是将分量密度 $\phi(y_j;\mu_i,\sum_i)$替换为 $f(y_j;\theta_{ij},k_i)$。M-Step 中 $\pi_i^{(k+1)}(i=1,\cdots,g-1)$的更新公式为：

$$\pi_i^{(k+1)}=\sum_{j=1}^{n}\tau_{ij}^{(k)}/n$$

此处

$$\tau_{ij}^{(k)}=\pi_i^{(k)}f(y_j;\theta_{ij}^{(k)},k_i^{(k)})\Big/\sum_{h=1}^{g}\pi_h^{(k)}f(y_j;\theta_{hj}^{(k)},k_h^{(k)})$$

$k_i^{(k+1)}$ and $\beta_i^{(k+1)}$ 的更新需要对下式进行迭代求解：

$$\sum_{j=1}^{n}\tau_{ij}^{(k)}\partial\log f(y_j;\theta_{ij},k_i)/\partial k=0$$
$$\sum_{j=1}^{n}\tau_{ij}^{(k)}\partial\log f(y_j;\theta_{ij},k_i)/\partial\beta_i=0 \qquad (5.14)$$

gamma 混合分布中的第 i 个 gamma 分量的密度函数为：

$$f(y_j;\mu_{ij},\alpha_i)=\frac{\left(\frac{\alpha_i}{\mu_{ij}}\right)^{\alpha_i}y_j^{(\alpha_i-1)}\exp\left(-\frac{\alpha_i}{\mu_{ij}}y_j\right)}{\Gamma(\alpha_i)}$$

形状参数 $\alpha_i>0$ 不依赖于解释变量。可以用对数链接建模线性预测器：

$$\eta_{ij}=h_i(\mu_{ij})=\log\mu_{ij}=\beta_i^T x_j$$

参见式(5.14)，证明 gamma 混合分布的 M-Step 需要求解分线性方程：

$$\sum_{j=1}^{n}\tau_{ij}^{(k)}\{1+\log\alpha_i-\log\mu_{ij}+\log y_j-y_j/\mu_{ij}-\psi(\alpha_i)\}=0$$

$$\sum_{j=1}^{n}\tau_{ij}^{(k)}(-1+y_j/\mu_{ij})\alpha_i x_j=0$$

此处 $\psi(r)=\{\partial\Gamma(r)/\partial r\}/\Gamma(r)$是 digamma 函数。

(6) 在 5.5 节里讨论了 IEM 算法，用 $\Psi^{(k+b/B)}$表示 Ψ 在第(k+1)次扫描上的第 b 次(b=1,…,B)迭代后的值。考察含 g 个分量的正态混合模型(5.4.1 节)：在第(k+1)次扫描上的第 b 次迭代中，对于第(b+1)块中的 y_j(b=0,…,B−1；i=1,…,g)，部分 E-Step 用 $\tau_i(y_j;\Psi^{(k+b/B)})$替换 z_{ij}。参见式(5.7)，用 $T_{iq,b+1}^{(k+b/B)}$表示第(b+1)块(b=0,…,B−1；q=1,2,3)的充分统计量的条件期望，例如：

$$T_{i1,b+1}^{(k+b/B)}=\sum_{j\in S_b}\tau_i(y_j;\Psi^{(k+b/B)})\quad(i=1,\cdots,g)$$

此处 S_b 是{1,…,n}的子集，它包含属于第(b+1)块(b=0,…,B−1)的那些 y_j 的下标。根据式(5.7)和式(5.8)，证明 IEM 算法的第(k+1)次扫描上的第(b+1)次迭代中的 M-Step 对所有 i 的 $(i=1,\cdots,g)\pi_i;\mu_i,\sum_i$ 进行更新：

$$\pi_i^{(k+(b+1)/B)}=T_{i1}^{(k+b/B)}/n$$

$$\mu_i^{(k+(b+1)/B)}=T_{i2}^{(k+b/B)}/T_{i1}^{(k+b/B)}$$

$$\sum_i^{(k+(b+1)/B)}=\{T_{i3}^{(k+b/B)}-T_{i1}^{(k+b/B)-1}T_{i2}^{(k+b/B)}T_{i2}^{(k+b/B)T}\}/T_{i1}^{(k+b/B)}$$

此处，对于所有的(i=1,…,g)和(q=1,2,3)，有：

$$T_{iq}^{(k+b/B)}=T_{iq}^{(k+(b-1)/B)}-T_{iq,b+1}^{(k-1+b/B)}+T_{iq,b+1}^{(k+b/B)}\tag{5.15}$$

式(5.15)右边的第一、二项分别在前次迭代和扫描时已经得到了。实践中 IEM 算法会在早期的一小部分迭代上使用标准 EM 算法以避免“早期分量饥饿”问题[26]，这时有：

$$T_{iq}^{(k)}=\sum_{b=1}^{B}T_{iq,b}^{(k)}\quad(i=1,\cdots,g;q=1,2,3)$$

(7) Ng 和 McLachlan [26]用 IEM 算法为正态混合提供了一种选择块数的简单指导。对于分量协方差矩阵是对角阵的情形(如示例 5.1)，他们建议 $B\approx n^{1/3}$。对于示例 5.1 所用 Iris 集，该值为 $B\approx(150)^{1/3}$，所以在 Iris 上可取 B=5 并用例 5.1 取的初始 Ψ 值来运行 IEM 算法。可验证：(a)所得最终估计和对数似然度近似于 EM 算法的结果；(b)IEM 算法需要的扫描次数少于 EM 算法而且每次扫描都能增加似然度，见参考文献[27]。

(8) Ng 和 McLachlan [28]将 ECM 算法用于训练混合专家(ME)网络[10, 12]。ME 网络包含若干称为专家网络的模块。这些专家网络在输入空间的每个区域里近似 y_j 的分布，它将输入 x_j 映射为输出 y_j，其条件密度为 $f_h(y_j|x_j;\theta_h)$，此处的 θ_h 是第 h 个(h=1,…,M)

专家网络的未知参数向量。门控网络提供了标量系数集 $\pi_h(x_j;\alpha)$对各种专家的贡献进行加权，α 是门控网络的未知参数向量。ME 网络的最终输出是诸专家网络产生的所有输出的加权求和：

$$f(y_j \mid x_j;\Psi) = \sum_{h=1}^{M} \pi_h(x_j;\alpha) f_h(y_j \mid x_j;\theta_h)$$

在 EM 算法的不完整数据框架下，我们引入指示变量 z_{hj}，该变量根据 y_j 是否属于第 h 个专家而取 1 或 0。证明 Ψ 在完整数据上的对数似然度为：

$$\log L_c(\Psi) = \sum_{j=1}^{n} \sum_{h=1}^{M} z_{hj} \{\log \pi_h(x_j;\alpha) + \log f_h(y_j \mid x_j;\theta_h)\}$$

根据 $\theta_h(h=1,\cdots,M)$，Q-方程可以被分解为包含两项的形式 $Q(\Psi;\Psi^{(k)}) = Q_\alpha + Q_\theta$，其中：

$$Q_\alpha = \sum_{j=1}^{n} \sum_{h=1}^{M} \tau_{hj}^{(k)} \log \pi_h(x_j;\alpha)$$

$$Q_\theta = \sum_{i=1}^{n} \sum_{h=1}^{M} \tau_{hj}^{(k)} \log f_h(y_j \mid x_j;\theta_h)$$

此处

$$\tau_{hj}^{(k)} = \pi_h(x_j;\alpha^{(k)}) f_h(y_j \mid x_j;\theta_h^{(k)}) \Big/ \sum_{r=1}^{M} \pi_r(x_j;\alpha^{(k)}) f_r(y_j \mid x_j;\theta_r^{(k)})$$

(9) ME 网络的门控网络的输出通常用多项 logit(或软最大)函数建模：

$$\pi_h(x_j;\alpha) = \frac{\exp(v_h^T x_j)}{1 + \sum_{r=1}^{M-1} \exp(v_r^T x_j)} \quad (h = 1,\cdots,M-1)$$

$$\pi_M(x_j;\alpha) = 1 \Big/ \left(1 + \sum_{r=1}^{M-1} \exp(v_r^T x_j)\right)$$

此处 α 包含 $v_h(h=1,\cdots,M-1)$中的元素。证明 M-Step 中，对于$(h=1,\cdots,M-1)$，$\alpha^{(k+1)}$的更新估计可以通过求解下面的方程得到：

$$\sum_{j=1}^{n} \left(\tau_{hj}^{(k)} - \frac{\exp(v_h^T x_j)}{1 + \sum_{r=1}^{M-1} \exp(v_r^T x_j)} \right) x_j = 0$$

这是一组非线性方程。第 h 个专家的非线性方程不仅依赖参数向量 v_h，还依赖 α 中的其他参数向量，这也就是说不能对每个参数向量 v_h 进行更新。IRLS 算法[12]对这些参数向量隐含使用了独立性假设，Ng 和 McLachlan [28]提出的 ECM 算法中，用计算更简单的针对 v_h 的(M−1)次的 CM-Step 取代。

(10) McLachlan 和 Chang[17]用基于混合模型的方法对混合数据进行聚类分析，观测数据包含连续型变量和类别型变量。设 Y_j 的 p 个变量中的 p_1 个是类别型，第 q 个类别型变量取 m_q 个离散值$(q=1,\cdots,p_1)$。使用基于位置模型的聚类算法[20]，p_1 个类别型变量被统一转变成一个单一的有 S 个取值的多项式随机变量 U，$S = \prod_{q=1}^{p_1} m_q$ 是 p_1 个类别型变量所表现出的不同模式(位置)的数量。令$(u_j)_s$表示第 j 个实例的第 s 个位置的标签$(s=1,\cdots,S;\ j=1,\cdots,n)$，如果 p_1 个类别型变量的实现对应于第 s 个模式则$(u_j)_s = 1$，否则$(u_j)_s = 0$。

位置模型进一步假设在条件$(u_j)_s=1$时，$(p-p_1)$个连续变量的条件分布是均值为μ_{is}协方差为$\sum_i$的正态分布，这对所有 S 个取值都是一样的。令p_{is}是给定归属于第 i 个混合分量条件下，$(u_j)_s=1$的条件概率$(s=1,\cdots,S;\ i=1,\cdots,g)$。参见 5.4.1 节，证明在 EM 算法的第$(k+1)$轮迭代中，更新估计为：

$$\pi_i^{(k+1)} = \sum_{s=1}^{S}\sum_{j=1}^{n}\delta_{js}\tau_{ijs}^{(k)} \Big/ n$$

$$p_{is}^{(k+1)} = \sum_{j=1}^{n}\delta_{js}\tau_{ijs}^{(k)} \Big/ \sum_{r=1}^{S}\sum_{j=1}^{n}\delta_{jr}\tau_{ijr}^{(k)}$$

$$\mu_{is}^{(k+1)} = \sum_{j=1}^{n}\delta_{js}\tau_{ijs}^{(k)} y_j^* \Big/ \sum_{j=1}^{n}\delta_{js}\tau_{ijs}^{(k)}$$

$$\sum_i^{(k+1)} = \sum_{s=1}^{S}\sum_{j=1}^{n}\delta_{js}\tau_{ijs}^{(k)}(y_j^* - \mu_{is}^{(k+1)})(y_j^* - \mu_{is}^{(k+1)})^T \Big/ \sum_{s=1}^{S}\sum_{j=1}^{n}\delta_{js}\tau_{ijs}^{(k)}$$

δ_{js}取 1 或 0 是根据$(u_j)_s$取 1 还是取 0，y_j^* 包含 y_j 中的连续变量，并且对于所有的$(s=1,\cdots,S;\ i=1,\cdots,g)$有：

$$\tau_{ijs}^{(k)} = \pi_i^{(k)} p_{is}^{(k)} \phi\left(y_j^*;\mu_{is}^{(k)},\sum_i^{(k)}\right) \Big/ \sum_{h=1}^{g}\pi_h^{(k)} p_{hs}^{(k)} \phi\left(y_j^*;\mu_{hs}^{(k)},\sum_h^{(k)}\right)$$

参 考 文 献

[1] A. Asuncion and D. J. Newman. UCI Machine Learning Repository. University of California, School of Information and Computer Sciences, Irvine, 2007. http://www. ics. uci. edu/ mlearn/ MLRepository. html.

[2] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximisation technique occurring in the statistical analysis of probabilistic functions of Markov process. *Annals of Mathematical Statistics*, 41: 164-171, 1970.

[3] D. B ohning, P. Schlattmann, and B. Lindsay. Computer-assisted analysis of mixtures (C. A. MAN): Statistical algorithms. *Biometrics*, 48: 283-303, 1992.

[4] J. G. Booth and J. P. Hobert. Maximizing generalized linear mixed model likelihoods with an automated Monte Carlo EM algorithm. *Journal of the Royal Statistical Society B*, 61: 265-285, 1999.

[5] N. A. Campbell and R. J. Mahon. Amultivariate study of variation in two species of rock crab of genus *Leptograpsus*. *Australian Journal of Zoology*, 22: 417-425, 1974.

[6] D. R. Cox and D. Hinkley. *Theoretical Statistics*. Chapman & Hall, London, 1974.

[7] H. Cramér. *Mathematical Methods of Statistics*. Princeton University Press, New Jersey, 1946.

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39: 1-38, 1977.

[9] C. Fraley and A. E. Raftery. Mclust: Software for model-based cluster analysis. *Journal of Classification*, 16: 297-306, 1999.

[10] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3: 79-87, 1991.

[11] M. Jamshidian and R. I. Jennrich. Standard errors for EM estimation. *Journal of the Royal Statistical Society B*, 62: 257-270, 2000.

[12] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6: 181-214, 1994.

[13] E. L. Lehmann and G. Casella. *Theory of Point Estimation*. Springer-Verlag, New York, 2003.

[14] B. G. Leroux and M. L. Puterman. Maximum-penalized-likelihood estimation for independent and Markov-dependent mixture models. *Biometrics*, 48: 545-558, 1992.

[15] T. A. Louis. Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society B*, 44: 226-233, 1982.

[16] G. J. McLachlan and K. E. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York, 1988.

[17] G. J. McLachlan and S. U. Chang. Mixture modelling for cluster analysis. *Statistical Methods in Medical Research*, 13: 347-361, 2004.

[18] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions (2nd edition)*. Wiley, New Jersey, 2008.

[19] G. J. McLachlan and D. Peel. Robust cluster analysis via mixtures of multivariate t-distributions. In *Lecture Notes in Computer Science*, pages 658-666. Springer-Verlag, Berlin, 1998. Vol. 1451.

[20] G. J. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, New York, 2000.

[21] G. J. McLachlan and D. Peel. Mixtures of factor analyzers. In P. Langley, editor, *Proceedings of the 17th International Conference on Machine Learning*, pages 599-606, San Francisco, 2000. Morgan Kaufmann.

[22] G. J. McLachlan, D. Peel, K. E. Basford, and P. Adams. The emmix software for the fitting of mixtures of normal and t-components. *Journal of Statistical Software*, 4: No. 2, 1999.

[23] X.-L. Meng and D. Rubin. Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, 80: 267-278, 1993.

[24] X.-L. Meng and D. A. van Dyk. The EM algorithm-an old folk song sung to a fast new tune. *Journal of the Royal Statistical Society B*, 59: 511-567, 1997.

[25] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355-368. Kluwer, Dordrecht, 1998.

[26] S. K. Ng and G. J. McLachlan. On the choice of the number of blocks with the incremental EM algorithm for the fitting of normal mixtures. *Statistics and Computing*, 13: 45-55, 2003.

[27] S. K. Ng and G. J. McLachlan. Speeding up the EM algorithm for mixture model-based segmentation of magnetic resonance images. *Pattern Recognition*, 37: 1573-1589, 2004.

[28] S. K. Ng and G. J. McLachlan. Using the EM algorithm to train neural networks: Misconceptions and a new algorithm for multiclass classification. *IEEE Transactions on Neural Networks*, 15: 738-749, 2004.

[29] D. Peel and G. J. McLachlan. Robust mixture modelling using the t distribution. *Statistics and Computing*, 10: 335-344, 2000.

[30] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77: 257-286, 1989.

[31] C. R. Rao. *Linear Statistical Inference and Its Applications (2nd edition)*. Wiley, New York, 1973.

[32] F.-P. Schelp, P. Vivatanasept, P. Sitaputra, S. Sormani, P. Pongpaew, N. Vudhivai, S. Egormaiphol, and D. Böhning. Relationship of the morbidity of under-fives to anthropometric measurements and community health intervention. *Tropical Medicine and Parasitology*, 41: 121-126, 1990.

[33] G. C. G. Wei and M. A. Tanner. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85: 699-704, 1990.

第 6 章　PageRank

Bing Liu, Philip S. Yu

6.1　引言
6.2　算法描述
6.3　一个扩展：Timed-PageRank
6.4　小结
6.5　习题
参考文献

6.1 引 言

Web 搜索获得了巨大成功，在其背后基于链接的排序方法可谓居功至伟。最有名的基于链接的排序算法当属 PageRank[1,7]，在某种程度上甚至可以说是它成就了 Google 搜索引擎的辉煌。鉴于 Google 取得的巨大商业成功，PageRank 也被视为主流的 Web 链接分析模型。

PageRank 算法是由 Sergey Brin 和 Larry Page 于 1998 年 4 月在第七届国际 WWW 大会(the 7th International World Wide Web Conference)上第一次提出的。在此前，搜索引擎采用的排序算法是基于内容的，简单地讲，搜索引擎主要的依据是用户查询和(被索引)网页之间的内容相似度，并根据这种相似度排序将相关网页返给用户。其实，这种做法是将传统信息检索的方法直接拿来实现，而没有针对性地考虑和挖掘 Web 检索任务的特性。从 1996 年开始，人们意识到以内容相似度为核心的搜索算法在两个重要的方面存在不足：第一方面，从 20 世纪 90 年代中晚期开始网页的数量迅速增长，几乎对于任何查询，相关网页的数量都是非常巨大的。例如，给定搜索查询“classification technique”，Google 搜索引擎就能检索到约 1000 万个相关网页，如此巨大的数量给排序造成了很大的困难——要如何从中选择 10 到 30 个网页？进而又如何排序展现给用户？第二方面，内容相似性方法很容易受到垃圾信息的困扰。编写页面的人为了提高网页的排名，通常会在网页中重复一些重要的词语，或者针对大量查询在网页中添加许多相关词语以使得该页面与许多查询高度相关。这些做法导致搜索结果垃圾化。

所以从 1996 年开始，无论是学术界还是搜索引擎公司都开始着手研究如何克服这些问题。研究中他们发现超链接非常有用。在传统信息检索中，集合中的文档被当做是相互独立的个体，但互联网的网页情况有所不同，他们是通过超链接联系起来的，而这些超链接蕴含着重要的信息。互联网上的链接可以分为两类：一类链接用来对站点的大量信息进行组织，所以这类链接就指向同一站点内的页面；另一类链接则指向其他站点的页面，这种外向型超链接起到一种向目标网页隐式传递权威性的作用。例如，如果你的网页指向一个外部的一个站点的网页，这显然表明你相信这个外部站点包含了对你有用的、有一定质量的信息。因此，那些被许多网页指向的网页，相对而言，就更可能包含了高权威性或说高质量的信息。显然，这些链接信息会非常有助于被搜索引擎评估和排序网页。PageRank 就是非常巧妙地利用这些链接形成的一个强有力的排名算法。从本质上讲，PageRank 依靠的是 Web 内在的民主性，该算法的关键是借助于整个 Web 的庞大链接结构去度量每个单独网页的质量。该算法把超链接形象地比作投票，比如一条从网页 x 到网页 y 的超链接可以认为是网页 x 投给了网页 y 一票。再深入地看，PageRank 不仅考虑网页获得的绝对票数(入链数目)，它还对投票的那些网页进行区别对待。本身“重要”的那些网页投出的票权重更大，那么得到它们投票的那些网页也就变得更加“重要”。这正是社会网络中排序声望(rank prestige)的核心思想[9]。本章，我们将主要介绍 PageRank 这个算法，附带地，我们也会提一下 Timed-PageRank 算法，它是标准 PageRank 算法的扩展，其主要特色是为搜索增加一个时间维度，这样可以有效处理 Web 信息的动态特性和老化过程。

6.2 算法描述

PageRank生成的Web网页排序是静态的，这是指每个网页的排序值是通过离线计算得到的，并且该值与查询无关。也就是说，网页排序值的计算纯粹基于Web上现有链接，而不考虑任何用户的任何查询。在讨论PageRank的公式之前，先阐述几个相关的主要概念：

网页i的入链(in-links)：那些指向网页i的来自于其他网页的超链接，通常不包括来自同一站点内网页的超链接。

网页i的出链(out-links)：那些从网页i指向其他网页的超链接，通常不包括连到同一站点内网页的超链接。

接下来我们阐述一些源于排序声望(rank prestige)[9]的思想，从而自然导出PageRank算法：

(1) 从一个网页指向另一网页的超链接是一种权威性的隐式传输，这样，网页i的入链越多，就表示它得到的声望越高。

(2) 指向网页i的网页也是有自己的声望分数。对于网页i来说，指向它的网页中，那些高声望网页比低声望网页更重要。换句话说，一个被其他重要网页指向的网页是重要的。

根据社会网络的排序声望原理，网页i的重要程度(PageRank值)由指向网页i的所有网页的PageRank值总和决定的。需要注意的是，一个网页可能会指向多个其他网页，该网页的声望分值就应该被它指向的所有页面分享。

为了将上述思想形式化表示，我们可以把Web抽象成一个有向图G=(V, E)，其中V是图的节点集合(节点对应网页)，E是图的有向边集合(有向边对应超链接)。设Web上的网页总数为n(即，n=|V|)。网页i的PageRank分值用P(i)表示：

$$P(i) = \sum_{(j,i)\in E} \frac{P(j)}{O_j} \tag{6.1}$$

O_j是网页j中出链的数量。此时，用数学的观点看就存在一个包含n个未知量的线性方程组，可以用一个矩阵来表示。首先作一个符号的约定，用黑体表示矩阵。矩阵**P**是表示PageRank值的n维行向量，如下：

$$\mathbf{P} = (P(1), P(2), \cdots, P(n))^T$$

再用矩阵**A**表示有向图的邻接矩阵，并按如下规则为每条有向边赋值：

$$A_{ij} = \begin{cases} \frac{1}{O_i} & \text{当}(i,j) \in E \\ 0 & \text{其他} \end{cases} \tag{6.2}$$

基于这两个矩阵，我们可以得到一个n维方程组：

$$\mathbf{P} = \mathbf{A}^T\mathbf{P} \tag{6.3}$$

这个等式是数学中的矩阵特征方程，该方程的解**P**是对应特征值为1的那个特征向量。因为这是一种循环定义，所以可以使用迭代算法进行求解。从数学上讲，如果满足一定的条件(会在后面简单提及)，那么PageRank向量的**P**就是对应矩阵**A**的最大特征值1的那个主特征向量。著名的幂迭代方法可以用来求**P**。

这里提到的条件为：**A** 应是一个随机矩阵且是不可约和非周期的。然而，Web 图并不满足这些条件。事实上，式(6.3)也可以从马尔科夫链理论导出，这样就能在此处使用马尔科夫链的一些理论成果，但是前提仍然是以上的三个条件。

可以将整个 Web 图用马尔科夫链进行建模，这样每一个网页(或者节点)可以被看做是马尔科夫链的一个状态，而超链接(或者有向边)表示状态转移，就是指马尔科夫链会以一定的概率从一个状态转到另一个状态。这样，该分析框架就将 Web 冲浪表示成一种随机过程——马尔科夫链，其状态转移的直观解释就是 Web 冲浪者的网上冲浪行为。

现在我们来考察一下 Web 图，看看为何它不能满足上述三个条件？我们首先来分析第一个条件是否成立？答案是不成立，即 **A** 不是随机(转移)矩阵。有限马尔可夫链的状态转移矩阵是随机矩阵，它要求每个元素都是非负实数，且每行加起来和为 1。这也就意味着要求每一个 Web 网页必须有至少一个出链。但实际中的 Web 不是这样的，很多网页根本没有出链，这种情况在状态转移矩阵 **A** 中表现为有些行全是由 0 组成的，这样的网页叫做悬挂网页。

例 6.2.1　图 6.1 展示了一个超链接图。

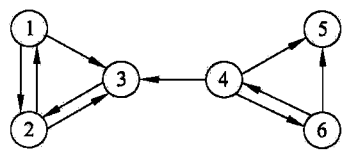


图 6.1　一个超链接图的例子

假设 Web 冲浪者会等概率随机点击网页上的超链接，那么从图 6.1 所示的超链接图就可以得到如下的一个状态转移矩阵：

$$\mathbf{A}=\begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix} \tag{6.4}$$

例如，因为节点 1 拥有两个出链(分别链向节点 2 和节点 3)，所以有 $\mathbf{A}_{12}=\mathbf{A}_{13}=1/2$。我们看到 **A** 的第 5 行全是 0，即网页 5 是一个悬挂网页，所以 **A** 不是随机矩阵。

那怎么办呢？我们可以从每一个悬挂网页 i 向每个网页引一条链接，这样就可以解决这个问题。再具体些，将网页 i 到每个网页的转换概率都设为 1/n，相当于均匀分布。也就是，把每一个全 0 的行都用全 **e**/n 来替换(**e** 是元素全为 1 的 n 维向量)，这样就得到以下矩阵：

$$\bar{\mathbf{A}}=\begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 1/3 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix} \tag{6.5}$$

在接下来的部分，我们都假设矩阵 **A** 已经完成这一修正，所以它是随机矩阵。

现在我们可以考察第二个条件，即 **A** 是不可约的吗？答案也是 No，因为 Web 图 G 不

是强连通的。

强连通图的定义是：有向图 G=(V, E)是强连通当且仅当对每一节点对 u,v∈V,存在从 u 到 v 的路径。

Web 图在一般情形下是可约的，因为图中通常存在一些节点对(u,v)，没有从 u 到 v 的路径。例如，在图 6.1 中，从节点 3 到节点 4 就没有路径。式(6.5)的调整不能确保图是不可约的。这个问题和下一个问题可以用一个策略处理(见下文)。

考察最后一个条件，即"A 是非周期的"也不成立。如果马尔科夫链中有周期的状态 i，那就意味着该链存在有向环路。

周期图的定义：说状态 i 是周期的并且具有周期 k>1，是指存在一个最小的正整数 k，使得所有从状态 i 出发又回到状态 i 的路径的长度都是 k 的整数倍。如果一个状态不是周期的(或者 k=1)，那它就是非周期的。如果一个马尔科夫链的所有状态都是非周期的，那么就说这个马尔科夫链是非周期的。

例 6.2.2 图 6.2 显示的是一个周期为 3 的马尔科夫链。转换矩阵 A 见图的左边。该链的每一个状态的周期都是 3，例如，如果从状态 1 开始，回到状态 1 的唯一路径是 1-2-3-1，如果走的次数为 h，回到状态 1 的路径需要的转移次数就是 3h。在 Web 中，类似的情形很多。

$$A=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 1 & 0 & 0\end{bmatrix}$$

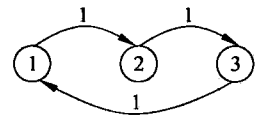


图 6.2 周期为 3 的马尔科夫链

这里，我们用一个的策略就可以轻松解决以上两个问题(可约、周期)：

从任一页面出发，到每个页面都加上一条链接，并给这一链接分配一个由参数 d 控制的微小转换概率。

经过这种修正的状态转换矩阵就成为不可约和非周期的了。这样我们可以得到一个改进的 PageRank 模型：

$$P\left((1-d)\frac{E}{n}+dA^{T}\right)P \tag{6.6}$$

此处的 E 表示 ee^{T}(e 是一个元素全为 1 的列向量)，所以 E 是一个 n×n 的元素全为 1 的矩阵。n 是 Web 图的节点总数，1/n 是跳转到一个随机网页的概率。式(6.6)中的 A 已经是随机矩阵了，该式经过化简得到：

$$P=(1-d)e+dA^{T}P \tag{6.7}$$

这样，任一网页 i 的 PageRank 公式为：

$$P(i)=(1-d)+d\sum_{j=1}^{n}A_{ji}P(j) \tag{6.8}$$

这和 PageRank 原始文献[1, 7]中的公式是一样的：

$$P(i)=(1-d)+d\sum_{(j,i)\in E}\frac{P(j)}{O_j} \tag{6.9}$$

参数 d，也称阻尼因子，可以在 0 到 1 范围内设定。文献[1, 7]中设置 d=0.85。

可以用幂迭代方法[2]来计算 Web 网页的 PageRank 值，该方法可以生成与特征值 1

对应的主特征向量。这个算法非常简单(见图 6.3),可以赋予任意的 PageRank 初始值。当 PageRank 值不再显著变化或者趋近收敛时,迭代算法就可结束。在图 6.3 中,差值向量的 1 阶范数小于一个预先指定的阈值 ε 后就结束算法迭代。

```
PageRank-Iterate(G)
    P_0←e/n
    k←1
    repeat
        P_k←(1−d)e+dA^T P_{k−1};
        k←k+1;
    until ||P_k−P_{k−1}||_1<ε
    return P_k
```

图 6.3 PageRank 的幂迭代求解算法

Web 搜索这种应用中,我们关注的主要还是网页排序。因此,并不追求非常严格的收敛,而是需要尽可能少的迭代次数。文献[1]报告了一个有 3.22 亿条链接的数据集上,该算法大概用 52 次迭代得到的结果就基本达到应用可以接受的程度。

自从 PageRank 的被提出后[1],研究者提出了很多强化模型、替代模型以及对其求解算法进行改进。Liu[5]和 Langville、Meyer[4]的书深入分析了 PageRank 和其他若干种基于链接的算法,例如另一种非常有名的算法 HITS[3]。

6.3 一个扩展:Timed-PageRank

经典的 PageRank 算法对搜索结果的时效性没有考虑和处理。但 Web 环境实际上是动态的,它处在持续的变化中。过去被认为高质量的网页在当前或者未来就未必还是高质量的。对搜索而言时效性其实是很重要的,因为用户通常都是更关注新信息。除了极少量既成事实和永恒经典能得以保持之外,占 Web 大部分比重的内容是经常变化的。我们看到在 Web 上,新网页或新内容是在不断增加。过时的东西则也应被摒弃,然而,实际上会有很多过时网页和链接不会被删掉。这会对搜索引擎带来麻烦,因为这些过时网页往往在过去长时间里积累了大量入链从而排序很高,而那些载有新信息的高质量新页面却因为它们没有足够的入链而排序偏低,这就会导致用户难以借助搜索引擎发现 Web 上的新信息。

Timed-PageRand 算法[6,10]在 PageRank 算法基础上增加了一个时间维度,它的思想其实很简单,主干上仍然是沿用 PageRank 的随机冲浪和马尔科夫链模型,不同之处在于 Timed-PageRank 不再使用常量阻尼因子 d,而是引入一个时间函数 $f(t)(0\leqslant f(t)\leqslant 1)$来"惩罚"陈旧的链接和网页(此处的 t 是指当前时间和网页上次更新时间的差值)。函数 f(t)被定义为网上冲浪者点击网页上链接进而前行的概率,而 $1-f(t)$就是网民不借助于网页上的链接而直接跳到一个随机选择的外部网页的概率。那么,对于一个特定的网页 i,网上冲浪者有两种选择:

(1) 以 $f(t_i)$的概率随机选择一个外链跳出。

(2) 以 $1-f(t_i)$的概率不通过链接跳到某个随机网页。

直观地讲,如果一个网页上一次更新(或创建)是发生在很久以前,那它所指向的网页就更老了,甚至是已经该废弃了。那么该网页对应的 $1-f(t)$值应该大,表示网上冲浪者更可能跳到一个随机网页。如果一个网页是新的,那对应的 $1-f(t)$值就应该小,表示网上冲浪者更可能选择该网页的出链前行。对于站点中没有入链的新网页,可以使用该站点过去网页的平均 TPR 值。这样做的合理性在于一个过去有质量的站点发布的新网页也应是有质

量的。Timed-PageRank 算法使用研究出版领域的检索语料进行了评估，性能表现优异。有兴趣的读者，可以研读文献[6]获得该算法的更多详细内容。

6.4 小 结

基于链接的排序已经成为 Web 搜索的关键技术。PageRank 是这方面最负盛名的算法，既在实践上非常有效，又有坚实的理论根据。本章只提供介绍性材料，更多细节可以参阅文献[1,4,5,7]。我们还简单介绍了一种能考虑时间维度的 PageRank 算法变体。最后，我们要强调的是基于链接的排序并不是搜索引擎使用的唯一排序策略，还需要结合许多传统信息检索和数据挖掘的方法，以及一些基于网页内容、用户点击等的启发式方法。

6.5 习 题

1. 给定如下的矩阵 **A**，请直接求解式(6.7)以获得 **P**。

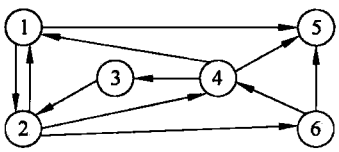
$$\mathbf{A}=\begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1/4 & 1/4 & 0 & 1/4 & 1/4 \\ 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix}$$

2. 给定习题 1 中的矩阵 **A**，请使用迭代求解方法计算 **P** 的前 10 次迭代。

3. 请计算习题 2 中每步迭代的平方错误，平方错误定义为 **P** 的每个分量的平方错误的总和。

4. 请画出习题 3 得到的平方错误曲线，以迭代次数作为 X 轴平方错误作为 Y 轴。请问：平方错误是否逐步下降？多少轮迭代后页面排序值才稳定下来？

5. 给定如下的图 **G**，请给出对一个的矩阵 **A**。



6. 对习题 5 中给出的图 **G**，请给出经过 7 轮幂迭代之后的 **P**。

7. 找一个 URL，收集以其为起点在最多 3 跳的距离上所遇到的所有网页，以此来构建一个 Web 图。

8. 请给出与习题 7 中得到的图对应的矩阵 **A**。

9. 请对习题 7 中得到的图计算前 7 轮幂迭代。

参 考 文 献

[1] S. Brin, and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30, 1998.

[2] G. H. Golub, and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1983.

[3] J. Kleinberg. Authoritative sources in a hyperlinked environment. *ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[4] A. N. Langville, and C. D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2006.

[5] B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data*. Springer, 2007.

[6] X. Li, B. Liu, and P. S. Yu. *Time Sensitive Ranking with Application to Publication Search*. Conference on Data Mining 2008.

[7] L. Page, S. Brin, R. Motwami, and T. Winograd. *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report 1999 0120, Computer Science Department, Stanford University, 1999.

[8] W. Steward. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.

[9] S. Wasserman, and K. Raust. *Social Network Analysis*. Cambridge University Press, 1994.

[10] P. S. Yu, X. Li, and B. Liu. Adding the Temporal Dimension to Search-A Case Study in Publication Search. WI-2005.

第 7 章 AdaBoost

Zhi-Hua Zhou，Yang Yu

7.1 引言
7.2 算法描述
7.2.1 符号定义
7.2.2 通用推举过程
7.2.3 AdaBoost 算法
7.3 示例
7.3.1 异或问题求解
7.3.2 真实数据上的性能
7.4 实际应用
7.5 高级主题
7.5.1 理论问题
7.5.2 多类别 AdaBoost
7.5.3 其他高级主题
7.6 软件实现
7.7 习题
参考文献

7.1 引 言

泛化能力处于机器学习的中心地位，它刻画了从给定训练数据集中学得的学习器处理未知新数据的能力。为了获得具有高度泛化能力的学习器，研究人员付出了巨大努力，其中最成功的一种泛型就是集成学习[32]。一般机器学习方法都是从训练数据中学得一个学习器，而集成学习要构建一组基学习器，并将它们进行集成。基学习器是通过决策树、神经网络及其他各种基学习算法从训练数据集习得的。所谓“众人拾柴火焰高”，集成学习的泛化能力一般明显优于单一学习器。实际上，集成学习最吸引人的就是它可以将稍优于随机猜测的弱学习器提升为预测精度很高的强学习器。通常假设基学习器是弱学习器。

1988 年 Kearns 和 Valiant 提出的一个有趣的问题“弱可学习和强可学习这两种复杂性类别是否等价”。如果二者等价就意味着性能略优于随机猜测的弱学习器可被“推举”为具有任意精度的强学习器！显然，这是一个对机器学习极为重要的问题。Schapire[21]发现答案是“Yes”，他给出的构造性证明成为最早的推举算法。该算法的一大缺点是要求预先知道基学习器的错误界，然而这在实际应用中常常是未知的。后来，Freud 和 Schapire 又提出了一种名为 AdaBoost[9]自适应推举算法，不再需要预先知道错误界等信息。显然，AdaBoost 理论意义重大，从而极大地推动了机器学习社区和统计学社区对集成学习方法的理论研究。值得一提的是，AdaBoost 论文[9]让 Schapire 和 Freund 赢得了 2003 年度理论计算机科学界最高奖——哥德尔奖(Godel Prize)，AdaBoost[9,10]也成为最有影响力的集成学习方法之一。

AdaBoost 及其变体由于理论基础坚实、预测准确和算法简单(Schapire 说只需 10 行代码)，所以被广泛应用于不同领域并获得了巨大成功。例如 Viola 和 Jones[27]用级联过程将 AdaBoost 和人脸检测算法结合，以矩形特征为弱学习器，利用 AdaBoost 加权弱学习器获得人脸检测所需的直观特征，此外还使用了一个级联过程(超出本章范围)来同时确保高性能和高精度，结果得到一个很强的人脸检测器：在一台 466MHz 的计算机上对一幅 384×288 像素图像进行人脸检测仅需 0.067s，这与当时最先进的人脸检测技术相比快了约 15 倍而且精度相当。这种人脸检测器被认为是上个十年中计算机视觉技术(特别是人脸检测技术)的一个最激动人心的突破。所以“推举”变成计算机视觉和很多其他应用领域的一个流行词了。

在本章的余下部分中，我们会介绍该算法及其实现，阐述该算法的运作过程。对于想更全面了解该算法的读者，我们将介绍一些理论成果和算法扩展的高级主题。

7.2 算法描述

7.2.1 符号定义

我们先介绍一些本章使用的符号。设$\mathcal{X}$为实例空间(或称特征空间)；$\mathcal{Y}$为待学习的隐含概念的标签集合，例如$\mathcal{Y}=\{-1,+1\}$表示二元分类；$\mathcal{D}$为包含 m 个实例的训练集，每个实例都关联了标签$\mathcal{D}=\{(x_i, y_i)\}$ $(i\in\{1,\cdots,m\})$。测试实例的标签是未知的需要被预测，我们

假定训练实例和测试实例都是从同一潜在分布$\mathcal{D}$独立抽样得到的。

学习算法$\mathcal{L}$从一个训练数据集$\mathcal{D}$训练后输出一个从$\mathcal{X}$到$\mathcal{Y}$的映射，称为假设 h 或者分类器。学习可以被看作是从假设空间中挑选最优假设的过程，这里的依据的是一个损失函数。分类任务对应于 0/1 损失函数：

$$\text{loss}_{0/1}(h \mid x) = I[h(x) \neq y]$$

I[·]是一个指示函数，即如果输入表达式正确则输出 1，否则输出 0，也就是说，如果错分一个实例就产生一个错误计数。本章默认使用 0/1 损失函数，但要知道推举算法也会使用其他损失函数。

7.2.2 通用推举过程

推举算法实际上是一族算法，AdaBoost 是其中最有影响的一个。所以，我们先从较为简单的一般推举过程谈起。

假设我们面临的是二元分类问题，即将实例分为正类和负类。我们通常假定存在一个未知的目标概念，它可以将那些属于该概念的实例标记为“正”，将其他实例标记为“负”。这个未知的目标概念就是我们期望学到的东西，我们可以叫它 ground-truth。对于二元分类问题，一个随机猜测分类器有 50%的 0/1 损失。

假如我们只有一个弱分类器，对于潜在实例分布$\mathcal{D}$，它的分类性能略优于随机猜测，比方说它有 49%的 0/1 损失，该弱分类器记为 h_1。显然 h_1 不是我们想要的，我们自然想到通过改正 h_1 所犯的错误来提高它。

我们可以试着从分布$\mathcal{D}$派生出分布$\mathcal{D}'$，在$\mathcal{D}'$上 h_1 的错误更显著，比如$\mathcal{D}'$主要由被 h_1 分错的实例构成(我们将在下一节介绍如何生成$\mathcal{D}'$)。这样，就可以用$\mathcal{D}'$训练得到分类器 h_2。但是 h_2 可能仍是个弱分类器。如果$\mathcal{D}'$满足一些条件，在$\mathcal{D}$中 h_1 表现欠佳的地方 h_2 会有比 h_1 更好的性能，同时在 h_1 表现好的地方 h_2 也能与之持平。所以，只要找到一种适当的方式集成 h_1 和 h_2(我们将在下一节介绍怎样集成)，得到的分类器就会比仅使用 h_1 的损失小。重复上述步骤，我们将会得到一个在$\mathcal{D}$上有非常小(理想情况是 0)的 0/1 损失的集成分类器。

简言之，推举算法顺序地训练一系列分类器，并将它们集成。分类器序列中后面的分类器更加关注前面分类器的错误。图 7.1 简述了一般的推举算法的流程。

AdaBoost

输入：实例分布$\mathcal{D}$；
　　基学习算法$\mathcal{L}$；
　　学习的循环次数 T
过程：
1. $\mathcal{D}_1 = \mathcal{D}$　　%初始分布
2. For t=1,…,T:
3. 　　$H_t = \mathcal{L}(\mathcal{D}_t)$；　　%从分布$\mathcal{D}_t$中训练弱学习器
4. $\epsilon_t = Pr_{x\sim\mathcal{D}_t,y} I[h_t(x)\neq y]$；　　%度量 h_t 的损失
5. $\mathcal{D}_{t+1} = \text{AdjustDistribution}(\mathcal{D}_t, \epsilon_t)$
6. End
输出：$H(x) = \text{CombineOutputs}(\{h_t(x)\})$

图 7.1　一般的推举算法程序

7.2.3　AdaBoost 算法

图 7.1 不是一个完整的算法，它还有一些待定的部分，譬如过程 AdjustDistribution 和 CombineOutputs。AdaBoost 算法可以看成是通用推举算法的一个实例化，算法概要参见图 7.2。

输入：数据集 $D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}$；
　　基学习算法 $\mathcal{L}$；
　　学习的循环次数 T
过程：
1. $\mathcal{D}_1(i)=1/m$　　%权重分布的初始化
2. for $t=1,\cdots,T$:
3. 　　$h_t=\mathcal{L}(D,\mathcal{D}_t)$;　　%从 D 中利用分布 $\mathcal{D}_t$ 训练学习器 h_t
4. 　　$\epsilon_t=Pr_{x\sim\mathcal{D}_t,y}I[h_t(x)\neq y]$　　%h_t 的错误度量
5. 　　if $\epsilon_t>0.5$, then break
6. 　　$\alpha_t=\frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$;　　%计算 h_t 的权重
7. 　　$\mathcal{D}_{t+1}(i)=\frac{\mathcal{D}_t(i)}{Z_t}\times\begin{cases}\exp(-\alpha_t) & \text{if } h_t(x_i)=y_i\\ \exp(\alpha_t) & \text{if } h_t(x_i)\neq y_i\end{cases}$　　%更新分布，Z_t 是归一化因子
　　$\frac{\mathcal{D}_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
8. end
输出：$H(x)=\mathrm{sign}\left(\sum_{t=1}^{T}\alpha_t h_t(x)\right)$

图 7.2　AdaBoost 算法

现在我们来介绍该算法的细节。AdaBoost 构造了一系列假设并对它们进行加权组合：

$$H(x)=\sum_{t=1}^{T}\alpha_t h_t(x)$$

从这里可以看到，AdaBoost 需要解决两个问题：即如何构造一系列假定 h_t；和如何确定合适的权重 α_t。

为了获得高效的错误消减过程，我们尝试最小化一个指数损失：

$$\mathrm{loss}_{exp}(h)=\mathbb{E}_{x\sim\mathcal{D},y}[e^{-yh(x)}]$$

$yh(x)$称为给定假设的分类间隔。

我们来考虑推举算法的一轮迭代。已经获得了一系列假设和它们的权重以及组合假设 H。现在，又有一个假定 h 与 H 合并为 $H+\alpha h$，相应的损失为：

$$\mathrm{loss}_{exp}(H+\alpha h)=\mathbb{E}_{x\sim\mathcal{D},y}[e^{-y(H(x)+\alpha h(x))}]$$

这个损失可以被分解到每一个实例上，称为逐点损失：

$$\mathrm{loss}_{exp}(H+\alpha h\mid x)=\mathbb{E}_{y}[e^{-y(H(x)+\alpha h(x))}\mid x]$$

这里的 y 和 h(x)取值是+1 或−1，该期望可以展开为：

$$\mathrm{loss}_{exp}(H+\alpha h\mid x)=e^{-yH(x)}(e^{-\alpha}\mid P(y=h(x)\mid x)+e^{\alpha}P(y\neq h(x)\mid x))$$

设我们已经得到 h，再令损失函数关于权重 α 的导函数为 0，即可最小化损失：

$$\frac{\partial\mathrm{loss}_{exp}(H+\alpha h\mid x)}{\partial\alpha}=e^{-yH(x)}(-e^{-\alpha}P(y=h(x)\mid x)+e^{\alpha}P(y\neq h(x)\mid x))=0$$

得到解：

$$\alpha=\frac{1}{2}\ln\frac{P(y=h(x)\mid x)}{P(y\neq h(x)\mid x)}=\frac{1}{2}\ln\frac{1-P(y\neq h(x)\mid x)}{P(y\neq h(x)\mid x)}$$

再解方程$\frac{\partial loss_{exp}(H+\alpha h)}{\partial\alpha}=0$，令$\epsilon=\mathbb{E}_{x\sim\mathcal{D}}[y\neq h(x)]$，我们可以得到

$$\alpha=\frac{1}{2}\ln\frac{1-\epsilon}{\epsilon}$$

这正是 AdaBoost 中计算 α_t 的方法。

我们接下来考虑如何得到 h。AdaBoost 通过调用一个基学习算法从一个特定的实例分布中生成一个假设。所以，我们只需考虑下一轮迭代需要什么样的假设，然后生成一个实例分布来构建这个假设。

我们可以将逐点损失扩展为关于 $h(x)=0$ 的二阶形式，固定 $\alpha=1$：

$$\begin{aligned}loss_{exp}(H+h\mid x)&\approx\mathbb{E}_y[e^{-yH(x)}(1-yh(x)+y^2h(x)^2/2)\mid x]\\&=\mathbb{E}_y[e^{-yH(x)}(1-yh(x)+1/2)\mid x]\end{aligned}$$

这里有 $y^2=1$ 和 $h(x)^2=1$。所以，一个好的假设是

$$\begin{aligned}h^*(x)&=\arg\min_h loss_{exp}(H+h\mid x)=\arg\max_h\mathbb{E}_y[e^{-yH(x)}yh(x)\mid x]\\&=\arg\max_h e^{-H(x)}P(y=1\mid x)\cdot1\cdot h(x)+e^{H(x)}P(y=-1\mid x)\cdot(-1)\cdot h(x)\end{aligned}$$

$e^{-yH(X)}$ 是常量。对期望部分进行归一化得到：

$$h^*(x)=\arg\max_h\frac{e^{-H(x)}P(y=1\mid x)\cdot1\cdot h(x)+e^{H(x)}P(y=-1\mid x)\cdot(-1)\cdot h(x)}{e^{-H(x)}P(y=1\mid x)+e^{H(x)}P(y=-1\mid x)}$$

改用 $w(x,y)$来表示期望，$w(x,y)$可以从 $e^{-yH(X)}P(y|x)$抽取：

$$h^*(x)=\arg\max_h\mathbb{E}_{w(x,y)\sim e^{-yH(x)}P(y|x)}[yh(x)\mid x]$$

$h^*(x)$取$+1$或-1，所以 $h^*(x)$和 $y|x$ 符号相同时得到最优解：

$$\begin{aligned}h^*(x)&=\mathbb{E}_{w(x,y)\sim e^{-yH(x)}P(y|x)}[y\mid x]\\&=P_{w(x,y)\sim e^{-yH(x)}P(y|x)}(y=1\mid x)-P_{w(x,y)\sim e^{-yH(x)}P(y|x)}(y=-1\mid x)\end{aligned}$$

可以看出，在分布 $e^{-yH(X)}P(y|x)$下得到 x 的最优分类。所以，$e^{-yH(X)}P(y|x)$是最小化 0/1 损失的理想假设。

所以，当假设 $h(x)$已经被学到，$\alpha=\frac{1}{2}\ln\frac{1-\epsilon}{\epsilon}$也已经在这一轮迭代中被定下来，那下一轮的分布就应该是

$$\mathcal{D}_{t+1}(x)=e^{-y(H(x)+\alpha h(x))}P(y\mid x)=e^{-yH(x)}P(y\mid x)\cdot e^{-\alpha yh(x)}=\mathcal{D}_t(x)\cdot e^{-\alpha yh(x)}$$

这就是 AdaBoost 算法更新实例分布的方法。

为什么对指数损失的优化就能最小化 0/1 损失呢？这是因为：

$$h^*(x)=\arg\min_h\mathbb{E}_{x\sim\mathcal{D},y}[e^{-yh(x)}\mid x]=\frac{1}{2}\ln\frac{P(y=1\mid x)}{P(y=-1\mid x)}$$

所以，我们有：

$$sign(h^*(x))=\arg\max_y P(y\mid x)$$

这意味着对于分类问题，指数损失的最优解达到了的最小贝叶斯错误。而且，最小化指数损失的函数 h^* 就是逻辑斯蒂回归模型再乘上一个因子 2。所以，忽略因子 1/2，AdaBoost

就可以看成加法逻辑斯蒂回归模型。

需要注意的是数据分布实际是未知的，而 AdaBoost 算法只能在有限实例的训练集上学习。所以，上述推导中的期望和权重的计算都只能基于训练实例。如果基学习器不能直接处理加权实例，可以用重采样机制来代替，该机制根据特定实例的权重来采样训练实例。

7.3　示　　例

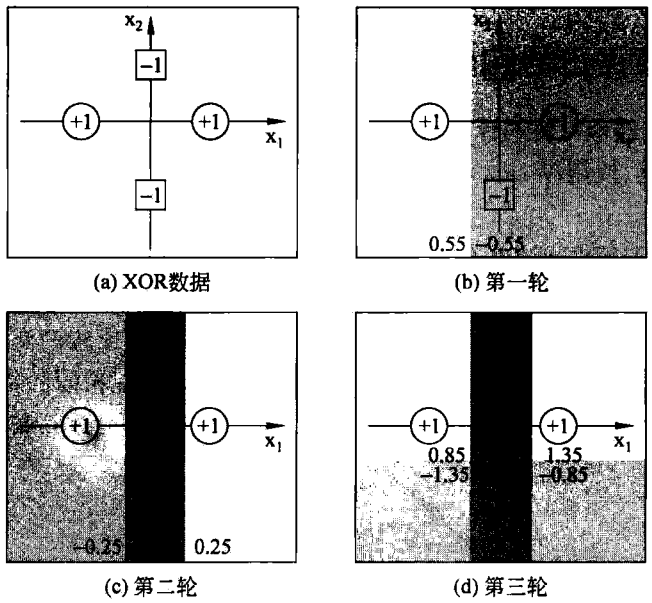
本节中，我们将演示 AdaBoost 算法是如何在玩具问题和真实数据集上工作的。

7.3.1　异或问题求解

我们考虑二维空间的一个仅包含 4 个实例的人工数据集，如图 7.3(a)所示。

$$\begin{Bmatrix}(\mathbf{x}_1=(0,+1),\mathrm{y}_1=+1)\\(\mathbf{x}_2=(0,-1),\mathrm{y}_2=+1)\\(\mathbf{x}_3=(+1,0),\mathrm{y}_3=-1)\\(\mathbf{x}_4=(-1,0),\mathrm{y}_4=-1)\end{Bmatrix}$$

用一条直线无法将两个类分开，这就是异或 XOR 问题。



(a) XOR数据　(b) 第一轮　(c) 第二轮　(d) 第三轮

图 7.3　AdaBoost 算法解 XOR 问题

假设我们有以下 8 个函数可供基学习算法选择，这 8 个函数都不够好。基学习算法面临性能一样的几个函数时会随机挑选一个。

$$\mathrm{h}_1(\mathbf{x})=\begin{cases}+1, & \text{当}(\mathbf{x}_1>-0.5)\\-1, & \text{其他}\end{cases}\qquad \mathrm{h}_2(\mathbf{x})=\begin{cases}-1, & \text{当}(\mathbf{x}_1>-0.5)\\+1, & \text{其他}\end{cases}$$

$$h_3(x)=\begin{cases}+1, & 当(x_1>+0.5)\\ -1, & 其他\end{cases} \quad h_4(x)=\begin{cases}-1, & 当(x_1>+0.5)\\ +1, & 其他\end{cases}$$

$$h_5(x)=\begin{cases}+1, & 当(x_2>-0.5)\\ -1, & 其他\end{cases} \quad h_6(x)=\begin{cases}-1, & 当(x_2>-0.5)\\ +1, & 其他\end{cases}$$

$$h_7(x)=\begin{cases}+1, & 当(x_2>+0.5)\\ -1, & 其他\end{cases} \quad h_8(x)=\begin{cases}-1, & 当(x_2>+0.5)\\ +1, & 其他\end{cases}$$

x_1 和 x_2 分别是 x 在第一、二维的值，现在看看 AdaBoost 算法是如何工作的：

(1) 第一轮是在原始数据上调用基学习算法。h_2、h_3、h_5 和 h_8 的分类错误是 0.25。设 h_2 被第一轮选为基学习器，实例 x_1 被分错。h_2 的权重是 $0.5\ln 3\approx 0.55$，如图 7.3(b)所示，阴影区域被分为负(−1)，分类是权重为 0.55 和−0.55。

(2) 第二轮增加 x_1 的权重并再次调用基学习算法。这轮 h_3、h_5 和 h_8 的分类错误相同。假设选中 h_3，其权重为 0.80。如图 7.3(c)所示，对 h_2 和 h_3 按权重组合，不同灰度级区分不同权重的负区域。

(3) 第三轮增加 x_3 的权重，此时 h_5 和 h_8 有相等的最小分类错误。设选中 h_5，其权重为 1.10。如图 7.3(d)所示，将 h_2、h_3 和 h_8 组合。从各区域的分类权重正负号看所有实例都被正确分类了。因此，AdaBoost 算法通过组合不完善线性分类器产生了一个零错误的非线性分类器。

7.3.2 真实数据上的性能

我们用 UCI 机器学习资料库中的 56 个数据集来评估 AdaBoost 算法，它覆盖了很多现实世界的任务。我们使用 Weka(将在 7.6 节介绍)的 AdaBoost.M1 算法对 50 个基学习器进行加权。

几乎所有的学习算法可以用作基学习算法，如决策树、神经网络，等等。在这里，我们尝试了 3 种，即决策树桩、剪枝和未剪枝 J4.8 决策树(Weka 的 C4.5 算法)。

比较结果参见图 7.4，每个圆圈代表一个数据集，并根据被比较的两个算法的预测错误来定位。在对角线上的圆圈表示的两个算法在相应数据集上有相同的错误。可以观察到，除了在少数偏置很强的数据集外，AdaBoost 算法的分类精度要胜过其基学习算法。

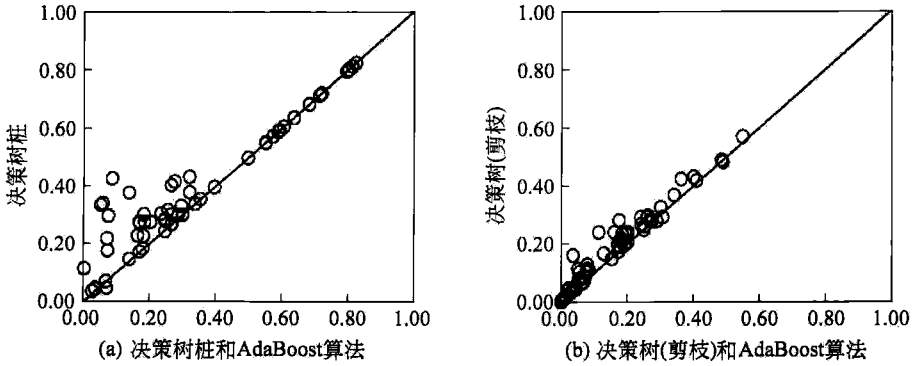
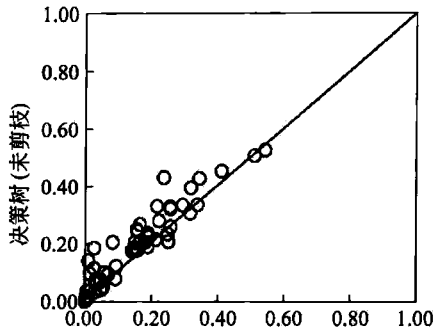


图 7.4 在 56 个 UCI 的数据集上，AdaBoost 算法和决策树桩，剪枝和未剪枝决策树的预测错误对比



(c) 决策树(未剪枝)和AdaBoost算法

图 7.4（续）

著名的偏置-方差分解[12]可阐明为什么 AdaBoost 有优异的性能[2,3,34]。这个强大的工具把预期学习错误分解为三个非负的量的总和，即固有噪声、偏差和方差。偏差度量的是学习方法对目标逼近程度的平均估计，方差度量的是学习方法在同样大小的不同训练集上波动程度的估计。从一些研究[2,3,34]可以观察到，AdaBoost 算法主要降低偏差，但它也能在一定程度上减少方差。

7.4 实际应用

Viola 和 Jones[27]用级联过程将 AdaBoost 算法和人脸检测器结合在一起。在一台 466MHz 机器中，该方法检测 384×288 像素的图像只需 0.067s，这几乎是当时最先进检测器的 15 倍，而且具有基本相同的检测精度。这种人脸检测器已被公认为在过去十年计算机视觉特别是人脸检测所取得的最令人兴奋的突破之一。在本节中，我们简要介绍 AdaBoost 算法是如何用到 Viola 和 Jones 的人脸检测器中的。

人脸检测是指在给定的图像中定位所有可能的人脸。首先，将图像划分成子图像（例如 24×24 个方块）。再将每个子图像表示为一个特征向量，因为要保证计算效率，所以要使用简单的特征。这里要考察子图像中所有可能的矩形，对每个矩形用图 7.5 所示的 4 个掩码来提取出 4 个特征。将掩码和矩形匹配，掩码白色区域对应的矩形像素灰度级之和减去掩码黑暗区域对应的矩形像素灰度级之和，得到的数值作为一个特征。这样，一个 24 × 24 图像划分会生成 100 万以上的特征，所以要求特征的计算速度非常快。

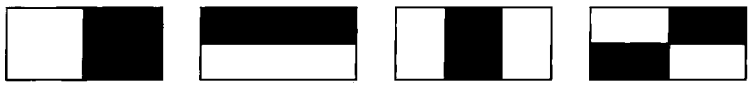


图 7.5 应用到每个矩形的 4 个特征掩码

将每个特征视为一个弱学习算法，也就是：

$$h_{i,p,\theta}(x) = I[px_i \leqslant p\theta] \quad (p \in \{+1,-1\})$$

其中 x_i 是 x 在第 i 个特征值。

这个基学习算法试图找到最小化分类错误的最好的弱分类器 h_i^*，P^*，θ^*，即

$$(i^*, p^*, \theta^*) = \underset{i,p,\theta}{\operatorname{argmin}} \mathbb{E}_{(x,y)} I[h_{i,p,\theta}(x) \neq y]$$

如图 7.6 所示含有人脸图像的矩形被当做正例，那些不包含任何人脸的矩形被当成反例。然后，运行 AdaBoost 算法得到一些弱学习器。每个弱学习器对应一个特征（共有 100 多万个特征）。从这个意义上讲，这里 AdaBoost 算法可以被视为一个特征选择工具。

图 7.6 训练集中的正例[27]

图 7.7 显示了排在最靠前的两个特征及其在人脸上的相对位置。这两个特征很直观，第一个特征度量眼睛区域和下边区域的密度差异，第二个特征度量眼睛区域和两只眼睛之间区域的密度差异。

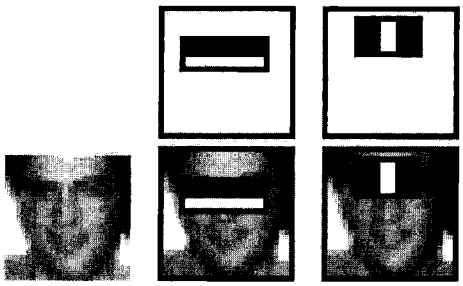


图 7.7 被选的特征[27]

用所选特征依序构建出一个非常不平衡的决策树，可称为级联分类器，如图 7.8 所示。

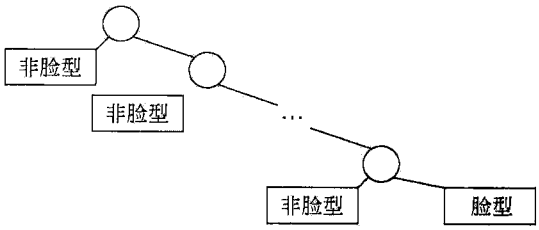


图 7.8 级联分类器

通过调整模型参数 θ，在每个树节点上，不是人脸的图像被分到“非人脸”分支，这样做极小化了假阴性。人脸检测中很容易将非人脸误判为人脸，而这种设计用一些特征过滤掉了很多的候选矩形，所以效率很高。据报道[27]，平均每个子图像仅用 10 个特征。图 7.9 显示了 Viola 和 Jones 检测器一些结果。

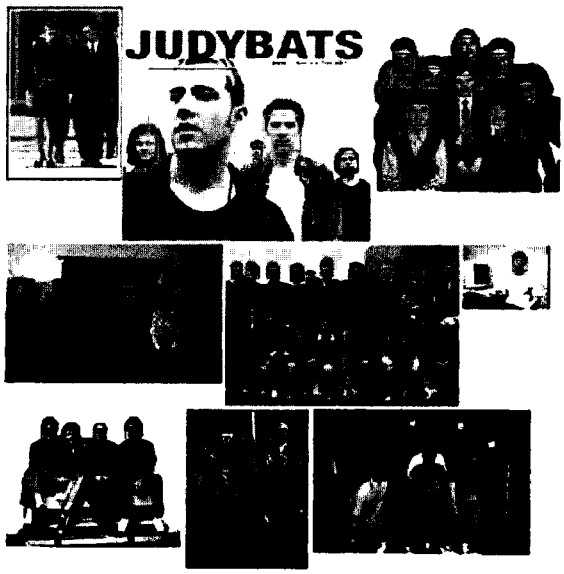


图 7.9　维奥拉-琼斯的人脸检测测试影像输出[27]

7.5　高级主题

7.5.1　理论问题

计算学习理论研究的是机器学习的一些基本理论问题。在这方面一个重要工作是 Valiant[25]于 1984 年提出的概率近似正确(PAC)框架，可对学习算法以分布无关的方式进行建模，粗略地讲，如果一个二元分类问题是可学习或强可学习的，是指存在一个算法多项式时间内输出一个假设 h，对于所有 $0<\delta,\varepsilon\leqslant 0.5$，有：

$$P(\mathbb{E}_{x\sim D,y}[I[h(x)\neq y]]<\varepsilon)\geqslant 1-\delta$$

而弱学习是指对所有 $0<\delta\leqslant 0.5$ 和略小于 0.5 的 ε，上式成立(h 只略强于随机猜测)。

1988 年，Kearns 和 Valiant [15]提出一个有趣而至关重要的问题“强学习问题类是否等价于弱学习问题类”。如果答案是“Yes”，则任何弱学习算法都可被提升为一个强学习算法。1989 年 Schapire [21]证明答案正是“Yes”，他的结构化证明也是第一个真正推举算法。一年后，Freund [7]开发了一种更有效的算法，但这两种算法都要求预先知道基学习器的错误界，但实际情况该信息往往是未知的。后来在 1995 年，Freund 和 Schapire [9]发展出了 AdaBoost 算法，实践证明该算法的效能和性能都很好。

Freund 和 Schapire 证明了，如果 AdaBoost 算法的基学习器的错误是 $\varepsilon_1,\varepsilon_2,\cdots,\varepsilon_T$，那最终的错误 ε 将以下式为上界：

$$\varepsilon = \mathbb{E}_{x\sim\mathcal{D},y} I[H(x) \neq y] \leqslant 2^T \prod_{t=1}^{T} \sqrt{\varepsilon_t(1-\varepsilon_t)} \leqslant e^{-2\sum_{t=1}^{T}\gamma_t^2}$$

其中 $\gamma_t = 0.5 - \varepsilon_t$。由此可知 AdaBoost 算法的错误将以指数速度递减。此外，我们还可以推出，为使得错误小于 ε，T 应该以下式为上界：

$$T \leqslant \left[\frac{1}{2\gamma^2}\ln\frac{1}{\varepsilon}\right]$$

这里假定 $\gamma = \gamma_1 = \gamma_2 = \cdots = \gamma_T$。

当然，AdaBoost 算法实际只能在训练数据 D 之上运行，所以有：

$$\varepsilon_D = \mathbb{E}_{x\sim D,y} I[H(x) \neq y]$$

故这里的错误其实是训练错误。而泛化错误关注的实际是实例分布$\mathcal{D}$上的错误：

$$\varepsilon_{\mathcal{D}} \leqslant \mathbb{E}_{x\sim\mathcal{D},y} I[H(x) \neq y]$$

早期的一些分析[9]表明，AdaBoost 算法的泛化错误是以下式为上界的：

$$\varepsilon_{\mathcal{D}} \leqslant \varepsilon_D + \tilde{O}\left(\sqrt{\frac{dT}{m}}\right)$$

至少以 $1-\delta$ 的概率成立。其中 d 是基学习器的 VC 维，m 是训练实例的数量，使用 $\tilde{O}(\cdot)$而非 $O(\cdot)$目的在于隐藏对数项和常量因子。

这个界意味着为了获得良好的泛化能力，应该限制基学习器的复杂度和学习的轮数，否则 AdaBoost 算法将过拟合。然而，经验却表明 AdaBoost 算法一般不会发生过拟合。也就是说，即便训练错误降到 0 或者进行了很多轮(比如 1000 轮)学习，测试错误仍然会继续降低。

例如，Schapier 等展示了 AdaBoost 算法在 UCI 机器学习资料库中的 letter 数据集上的性能。参见图 7.10(左)，其中较高的曲线是测试错误，较低的曲线是训练错误。AdaBoost 在不超过 10 轮学习后达到了零训练错误而泛化错误继续降低。这个现象似乎与机器学习的基本原理之一的“Occam 剃刀原理”相矛盾，该原理说不应做超过必需要做的事情。

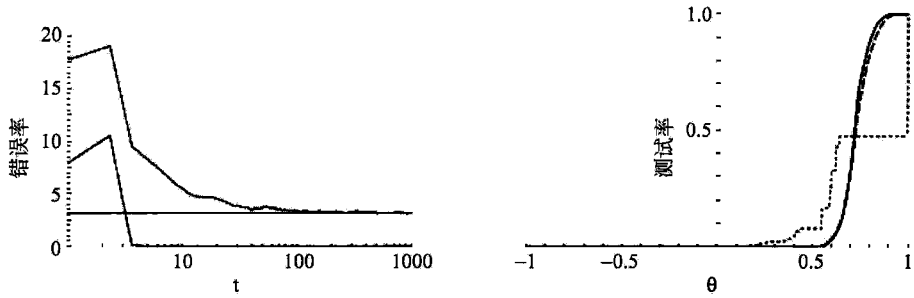


图 7.10 训练和测试的错误率(左)；AdaBoost 在 letter 数据集上间隔分布(右[22])

很多人对此进行了研究并提出一些理论解释[11]，如 Schapire 等[22]提出了基于间隔的解释，他们证明 AdaBoost 算法能够在训练错误减少为零之后间隔还能继续增大，所以就算是很多轮学习后也不会产生过拟合。h 在 x 上的分类间隔被定义为 yh(x)，而 $h(x) = \sum_{t=1}^{T}\alpha_t h_t(x)$ 的间隔则被定义为：

$$yh(x) = \frac{\sum_{t=1}^{T}\alpha_t y h_t(x)}{\sum_{t=1}^{T}\alpha_t}$$

图 7.10(右)是 $yH(x)\leqslant\theta$ 在不同 θ 值上的分布情况。已经证明[22],泛化错误的上界是:

$$\varepsilon_{\mathcal{D}} \leqslant P_{x\sim D,y}(yH(x)\leqslant\theta)+\tilde{O}\left(\sqrt{\frac{d}{m\theta^2}+\ln\frac{1}{\delta}}\right)$$

$$\leqslant 2^T\prod_{t=1}^{T}\sqrt{\varepsilon_t^{1-\theta}(1-\varepsilon)^{1+\theta}}+\tilde{Q}\left(\sqrt{\frac{d}{m\theta^2}+\ln\frac{1}{\delta}}\right)$$

以大于或等于 $1-\delta$ 的概率成立。当上界中的其他变量固定时,泛化错误随着间隔的增大而降低。

Brieman 提出了有更紧致泛化错误上界的最小间隔 $e=\min_{x\in D} yH(x)$[4],他给出了证明。正是受到这一紧间隔理论的启发,人们提出了 arc-gv 算法(AdaBoost 算法的一个变体),该算法直接最大化最小间隔,方法是更新 α_t:

$$\alpha_t=\frac{1}{2}\ln\left(\frac{1+\gamma_t}{1-\gamma_t}\right)-\frac{1}{2}\ln\left(\frac{1+e_t}{1-e_t}\right)$$

虽然 arc-gv 算法的最小间隔一致优于 AdaBoost 算法,但所有测试数据上 arc-gv 算法的测试错误都急剧增加,这一矛盾让 AdaBoost 的间隔理论看起来几乎死定了。

泛化错误的上界和间隔、学习轮数以及基学习器的复杂度相关。2006 年,Reyzin 和 Schapire[20]报告了一个有趣的发现:他们发现 Breiman 在比较 arc-gv 算法和 AdaBoost 算法性能时,为了控制基学习器的复杂度而采用了具有相同叶节点数目的决策树,这些树的形状差异很大,Arc-gav 算法产生的决策树有着比较大的深度,而 AdaBoost 算法产生的决策树有比较大的宽度。如图 7.11(左)显示了 UCI 机器学习资料库的 breast cancer 数据集上,这两种算法所产生的决策树的深度的差别。虽然两者有同样的叶子数目,但较深的树有更多的属性测试,因此二者有不同的复杂度。所以,Reyzin 和 Schapire 用 decision stump 重做了 Breiman 的实验,这个 decision stump 只有一个叶子(因此其复杂度固定),这时看到 AdaBoost 的间隔分布实际上比 arc-gv 算法的要好,如图 7.11(右)所示。

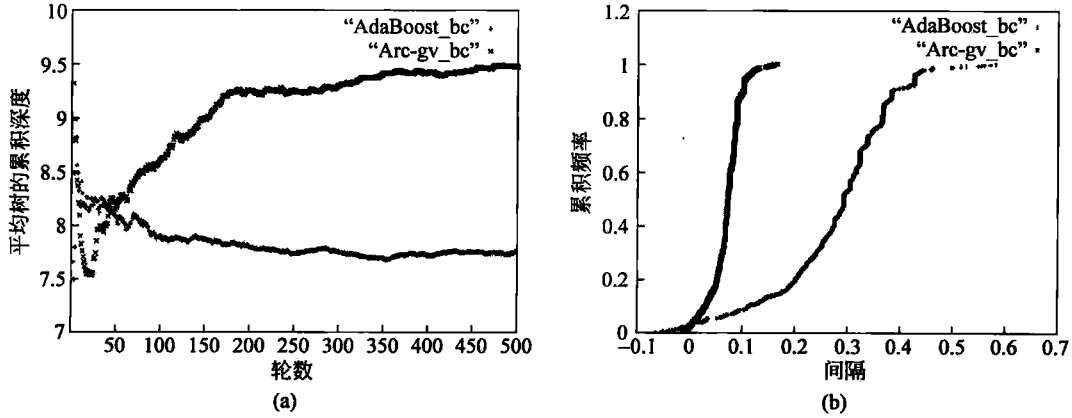


图 7.11 AdaBoost 和 arc-gv 的树深度(左图)和边界分布(右图)的比较

最近,Wang 等提出了均衡间隔理论,证明它的上界比最小间隔的上界还要紧致,这也表明最小间隔对于 AdaBoost 算法未必关键。这也导致一个开放问题:是否可以发明一种

直接最大化均衡间隔的算法，再来观察这种算法的测试误差是否比 AdaBoost 算法的小。

7.5.2 多类别 AdaBoost

在上面的内容里我们仅关心二元分类上的 AdaBoost 算法，即 $Y=\{+1,-1\}$。对于多类别任务，实例可以属于多类别中的一个(而不限于两个类别中的一类)。比如，每个手写数字属于 10 个类中的一个，即 $Y=\{0,\cdots,9\}$。处理多类分类问题的方法很多。

AdaBoost. M1 采取了一种对图 7.2 所示算法的非常直接的扩展，基学习器是多类学习器而不是二元分类器。该算法不能使用二元基分类器，并且要求每个基学习器都有低于 1/2 的多类 0/1 损失，但这个约束过强了。

SAMME[35]是对 AdaBoost. M1 的改进，它将图 7.2 中 AdaBoost. M1 算法的第五行替换为：

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)+\ln(|Y|-1)$$

这个修改是从最小化多类指数级损失推导而来的。已证明，和二元分类的情况相似，优化多类指数损失会达到最优贝叶斯错误，也就是：

$$\text{sign}[h^*(x)] = \underset{y\in Y}{\text{argmax}} P(y \mid x)$$

其中 h^* 是多类指数损失的最优解。

另一种常见的解决多类分类问题的方法是，将任务分解为多个二元分类问题。常见直接分解方法包括“一对其他”和“一对一”。“一对其他”是将一个含 $|Y|$ 个类别的多类任务分解为 $|Y|$ 个二元分类任务，其中第 i 个任务要确定一个实例是否属于第 i 类。一对一是将含 $|Y|$ 个类别的多类任务分解为 $\frac{|Y|(|Y|-1)}{2}$ 个二元分类任务，其中每个任务要确定一个实例是属于第 i 类还是第 j 类。

AdaBoost. MH[23]采取的是一对其他方法。在训练完 $|Y|$ 个二元 AdaBoost 分类器之后，每个 AdaBoost 分类器的实数型输出 $H(x)=\sum_{i=1}^{T}\alpha_t h_t(x)$ 被用来寻找最有可能的类别，即

$$H(x) = \underset{y\in Y}{\text{argmax}} H_y(x)$$

其中 H_y 表示将第 y 类从其他类中分离出来的 AdaBoost 分类器。

AdaBoost. M2[9]算法采取的是一对一方法，这个方法最小化伪损失。这个算法后来被推广成 AdaBoost. MR[23]，这个算法最小化的是排序损失，这是因为通常认为排序靠前的类最有可能是正确的类。

纠错输出编码(ECOC)[6]也能将多类分类问题分解为一系列二元分类问题。比如，图 7.12(a)演示了“4 个类别 5 个分类器”的输出编码。每个分类器根据对应的列被训练为判别+1 和−1 的类别。测试实例是要综合 5 个分类器的输出，这样可以得到一个预测编码向量。将该向量与类别编码向量(图 7.12(a)的每一行)使用海明距离比较，有最短距离的类就被当成最终预测的结果。根据信息论，对于独立的二元分类器，编码向量的最小海明距离越大，0/1 损失越小。据此提出了一个多类分解方法的统一框架。图 7.12(b)显示的是一

对其他分解方法的输出编码，图 7.12(c)显示的是一对一分解方法的输出编码，其中的 0 表示分类器应该忽略那些类别的实例。

$H_1\ H_2\ H_3\ H_4\ H_5$	$H_1\ H_2\ H_3\ H_4$	$H_1\ H_2\ H_3\ H_4\ H_5\ H_6$
y_1=+1−1+1−1+1	y_1=+1−1−1−1	y_1=+1+1+1 0 0 0
y_2=+1+1−1−1−1	y_2=−1+1−1−1	y_2=−1 0 0+1+1 0
y_3=−1−1+1−1−1	y_3=−1−1+1−1	y_3= 0−1 0−1 0+1
y_4=−1+1−1+1+1	y_4=−1−1−1+1	y_4= 0 0−1 0−1−1
(a) 初始码	(b) 一对其他码	(c) 一对一码

图 7.12　ECOC 的输出编码

7.5.3　其他高级主题

对用户来说，学习模型的可理解性在很多实际应用中都是非常重要的。和其他集成学习方法相似，AdaBoost 及其变种的一个严重缺陷就是缺乏可理解性。甚至当基学习器是可理解的模型(如小的决策树)，但它们集成后就成了黑盒模型。所以提高集成学习方法的可理解性是一个非常重要的研究方向[33]。

绝大多数的集成学习方法会将所有生成的基学习器进行集成。但是，研究表明可以通过选择集成的方法用更少量的基学习器得到强集成学习器[34]。早期的一些研究[17，24]表明对集成算法的裁剪可能会损害泛化性能，所以就要探索好的选择或裁剪算法[18，31]。

在许多应用中，会出现某个类的训练样本远多于别的类。如果不考虑类不平衡性整个学习算法就容易被大类主导。但是，人们关心的却往往是小类。AdaBoost 算法发展了许多处理类不平衡问题的变体[5,14,19,26]。近期的研究[16]表明，可用 AdaBoost 算法判断任务是否存在类不平衡问题。

如前文所说，除了 0/1 损失之外，推举算法还可以用其他损失函数。例如，通过构造排序损失函数，RankBoost [8]和 AdaRank [30]已经被用到信息检索中。

7.6　软 件 实 现

AdaBoost 是最先进的机器学习算法之一，该算法及其变体的实现很容易获取，而且有 Java、MATLAB、R 和 C++ 等编码版本。

AdaBoost 的 Java 版本可以在 Weka 找到。Weka 是著名的机器学习和数据挖掘的开源包，它的 AdaBoost. M1 算法[9]提供一些选项，用户可以选择基学习算法、设置基学习器的数量以及允许在重加权和重采样上切换。Weka 还包括了其他推举算法，比如 LogitBoost[11]、MultiBoosting [29]等。

Spider 提供了 AdaBoost 的 MATLAB 实现；R-Project 提供了 AdaBoost 的 R 实现。也可以在 Sourceforge 上找到 AdaBoost 的 C++ 实现。当然，你可以在互联网上找到很多其他实现。

7.7 习 题

1. 请叙述 Boosting 的基本思想。

2. 图 7.2 中，为什么在 $\varepsilon_t \geqslant 0.5$ 时跳出循环？

3. 给定如下的训练集：

$$\left\{\begin{array}{l}(x_1=(+1,0),y_1=+1)\\(x_2=(0,+1),y_2=+1)\\(x_3=(-1,0),y_3=+1)\\(x_4=(0,-1),y_4=+1)\\(x_5=(0,0),y_5=-1)\end{array}\right\}$$

请问是否存在实现零训练错误的线性分类器？为什么？

4. 对于习题 3 中的训练集，请验证 AdaBoost 算法用如下的 5 个线性分类器就能实现零训练错误：

$$h_1(x)=2I[x_1>0.5]-1 \qquad h_2(x)=2I[x_1<0.5]-1$$
$$h_3(x)=2I[x_1>-0.5]-1 \qquad h_4(x)=2I[x_1<-0.5]-1$$
$$h_5(x)=2I[x_2>0.5]-1 \qquad h_6(x)=2I[x_2<0.5]-1$$
$$h_7(x)=2I[x_2>-0.5]-1 \qquad h_8(x)=2I[x_2<-0.5]-1$$
$$h_9(x)=+1 \qquad h_{10}(x)=-1$$

5. 在上面的习题 4 中 AdaBoost 算法能否在 $T \geqslant 5$（T 是基础分类器的数量）时达到零训练错误。

6. 最近邻分类器使用与目标示例最近的训练样本的标签对目标实例进行分类。试问 AdaBoost 是否能提升此类分类器的性能？为什么？

7. 请将下列函数画在同一个图上，定义域都取 $z \in [-2, 2]$，请观察它们的不同。

$$l_1(z)=\begin{cases}0, & z\geqslant 0\\1, & z<0\end{cases} \qquad l_2(z)=\begin{cases}0, & z\geqslant 1\\1-z, & z<1\end{cases}$$
$$l_3(z)=(z-1)^2 \qquad l_4(z)=e^{-z}$$

当 $z=yf(x)$，函数 l_1、l_2、l_3、和 l_4 分别称为 0/1 损失、hinge 损失（用于支持向量机）、平方损失（用于最小均方回归）和指数损失（用于 AdaBoost）。

8. 习题 7 中的函数 l_2、l_3、和 l_4 都是凸函数（l 是凸函数当且仅当 $\forall z_1, z_2: l(z_1+z_2) \geqslant (l(z_1)+l(z_2))$）。考虑一个二分类任务 $z=yf(x)$，其中 $y=\{-1,+1\}$，请找到最优解为贝叶斯最优解的函数。

9. 请问 AdaBoost 是否能被扩展用来求解回归问题？如果可以那该如何做？如果不可以，原因是什么？

10. 请通过实验对 AdaBoost 使用重加权和重采样的效果进行比较，可以使用 Weka 以及 UCI 机器学习资料库提供的数据集。

参 考 文 献

[1] E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1: 113-141, 2000.

[2] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1-2): 105-139, 1999.

[3] L. Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, University of California, Berkeley, 1996.

[4] L. Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11 (7): 1493-1517, 1999.

[5] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer. SMOTEBoost: Improving prediction of the minority class in boosting. In *Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 107-119, Cavtat-Dubrovnik, Croatia, 2003.

[6] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via errorcorrecting output codes. *Journal of Artificial Intelligence Research*, 2: 263-286, 1995.

[7] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2): 256-285, 1995.

[8] Y. Freund, R. Iyer, R. E. Schapire, andY. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4: 933-963, 2003.

[9] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1): 119-139, 1997.

[10] Y. Freund and R. E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5): 771-780, 1999.

[11] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting (with discussions). *The Annals of Statistics*, 28(2): 337-407, 2000.

[12] S. German, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1): 1-58, 1992.

[13] T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(2): 451-471, 1998.

[14] M. V. Joshi, R. C. Agarwal, and V. Kumar. Predicting rare classes: Can boosting make any weak learner strong? In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 297-306, Edmonton, Canada, 2002.

[15] M. Kearns and L. G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 433-444, Seattle, WA, 1989.

[16] X.-Y. Liu, J.-X. Wu, and Z.-H. Zhou. Exploratory under-sampling for classimbalance learning. *IEEE Transactions on Systems, Man and Cybernetics-Part B*, 2009.

[17] D. Margineantu and T. G. Dietterich. Pruning adaptive boosting. In *Proceedings of the 14th International Conference on Machine Learning*, pages 211-218, Nashville, TN, 1997.

[18] G. Mart'inez-Mu~noz and A. Su'arez. Pruning in ordered bagging ensembles. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 609-616, Pittsburgh, PA, 2006.

[19] H. Masnadi-Shirazi and N. Vasconcelos. Asymmetric boosting. In *Proceedings of the 24th International Conference on Machine Learning*, pages 609-619, Corvallis, OR, 2007.

[20] L. Reyzin and R. E. Schapire. How boosting the margin can also boost classifier complexity. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 753-760, Pittsburgh, PA, 2006.

[21] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2): 197-227, 1990.

[22] R. E. Schapire, Y. Freund, P. Bartlett, andW. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5): 1651-1686, 1998.

[23] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence rated predictions. *Machine Learning*, 37(3): 297-336, 1999.

[24] C. Tamon and J. Xiang. On the boosting pruning problem. In *Proceedings of the 11th European Conference on Machine Learning*, pages 404-412, Barcelona, Spain, 2000.

[25] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11): 1134-1142, 1984.

[26] P. Viola andM. Jones. Fast and robust classification using asymmetric AdaBoost and a detector cascade. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1311-1318. MIT Press, Cambridge, MA, 2002.

[27] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2): 137-154, 2004.

[28] L. Wang, M. Sugiyama, C. Yang, Z.-H. Zhou, and J. Feng. On the margin explanation of boosting algorithm. In *Proceedings of the 21st Annual Conference on Learning Theory*, pages 479-490, Helsinki, Finland, 2008.

[29] G. I. Webb. MultiBoosting: A technique for combining boosting and wagging. *Machine Learning*, 40(2): 159-196, 2000.

[30] J. Xu and H. Li. AdaRank: A boosting algorithm for information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 391-398, Amsterdam, The Netherlands, 2007.

[31] Y. Zhang, S. Burer, and W. N. Street. Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research*, 7: 1315-1338, 2006.

[32] Z.-H. Zhou. Ensemble learning. In S. Z. Li, editor, *Encyclopedia of Biometrics*. Springer, Berlin, 2008.

[33] Z.-H. Zhou, Y. Jiang, and S.-F. Chen. Extracting symbolic rules from trained neural network ensembles. *AI Communications*, 16(1): 3-15, 2003.

[34] Z.-H. Zhou, J. Wu, andW. Tang. Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(1-2): 239-263, 2002.

[35] J. Zhu, S. Rosset, H. Zou, and T. Hastie. Multi-class AdaBoost. Technical report, Department of Statistics, University of Michigan, Ann Arbor, 2006.

第 8 章　kNN：k-最近邻

Michael Steinbach，Pang-Ning Tan

8.1　引言
8.2　算法描述
　　8.2.1　宏观描述
　　8.2.2　若干议题
　　8.2.3　软件实现
8.3　示例
8.4　高级主题
8.5　习题
致谢
参考文献

8.1 引 言

“死记硬背”分类器应该是最简单最初级的分类器了，它将全部训练数据都记下来，当测试对象的属性与某个训练对象的属性完全匹配时就可以对其进行分类。这种方法最显而易见的一个问题是，有的测试对象可能与任何训练对象都不能确切匹配从而不能对其进行分类。此外，这种方法遇到的另外一个问题是可能会遇到两个或者两个以上具有相同属性的训练对象却具有不同的类别标记。

比这个方法稍复杂一些的是 k-最近邻(kNN)分类法[10,11,21]。kNN 方法是指从训练集找出 k 个最接近测试对象的训练对象，再从这 k 个训练对象中找出居于主导的类别，将其赋给测试对象。所以，这种分类的方法可以有效地避开上面提到的分类任务中两个常见的问题：第一，在很多数据集中，一个对象和另一个对象完全匹配基本是不可能的(kNN 计算对象间距离，不进行对象匹配)；第二，具有相同属性的对象有不同的类别标记(kNN 依据总体占优的类别进行决策，不是根据单一对象的类别决策)。kNN 方法需要考虑几个关键要素：(1)用于决策一个测试对象类别的已被标记对象集合，(2)用来计算对象间临近程度的距离或其他相似性指标，(3)最近邻的个数 k，(4)基于 k 个最近邻及其类别来判定目标对象类别的方法，最简单的做法就是将离目标对象最近的一个邻居的类别赋给目标对象，也可以从邻居中选择占多数的那个类别，当然还有多种不同的改进方法，我们下面会讨论到。

从更普遍的意义上讲，kNN 是一种基于实例的学习方法[1](基于案例的推理也是一种基于实例的学习方法，只不过主要处理的是符号数据)。同时，kNN 也是一种惰性学习方法，就是说直到回答查询(预测阶段)时才去处理训练数据。

kNN 分类方法很容易理解和实现，而且它在许多情况下都表现非常良好。Cover 和 Hart[6]的研究表明，在一定的条件下，最近邻规则的分类错误率最多不会超过最优贝叶斯错误率的两倍，更进一步，一般情况下 kNN 方法的错误率都会渐进收敛到贝叶斯错误率，所以可以将 kNN 方法用作贝叶斯的近似。

由于 kNN 很简单，所以很容易对它进行改进而用于处理更复杂的分类问题。例如，kNN 特别适合多模分类问题和多标签分类问题。举一个多标签分类任务的例子：在基于微阵列表达的基因功能分配研究中，研究人员发现 kNN 要优于支持向量机(SVM，一种比 kNN 复杂得多的分类方法)[17]。

本章余下的部分将会介绍基本的 kNN 算法(即包括影响分类性能和计算性能的各种问题)，介绍一些 kNN 的软件实现，提供一个使用 Weka 机器学习包来进行最近邻分类的示例，再讨论一些高级主题，最后列一些习题。

8.2 算法描述

8.2.1 宏观描述

算法 8.1 是 kNN 分类方法的一个高度概括。给定一个训练集 D 和一个测试对象 z，该

测试对象是一个由属性值和一个未知的类别标签组成的向量，该算法需要计算 z 和每个训练对象之间的距离（或相似度），这样就可以确定最近邻的列表。然后，将最近邻中实例数量占优的类别赋给 z，当然也不是只能采取这一种策略，例如，甚至可以从训练集中随机选择一个类或选择最大类。

该算法的存储复杂度是 O(n)，其中 n 为训练对象的数量，时间复杂度也是 O(n)，因为需要计算测试对象和每个训练对象间的距离，值得注意的是，该算法不需要花费任何时间做模型的构造。与此不同的是，其他的大多数分类方法，如决策树或分类超平面等通常都需要一个模型构造的阶段，而且该阶段极其耗时，但它们在分类时非常省时，一般仅需要常数级的复杂度 O(1)。

算法 8.1　基本的 kNN 算法

输入：D，是训练集；z，测试对象，它是属性值构成的向量；L，对象的类别标签集合
输出：c_z 属于 L，即 z 的类别
foreach y 属于 D do
计算 d(y,z)，即 y 和 z 的距离；
end
从数据集 D 中选出子集 N，N 包含 k 个距 z 最近的训练对象
$c_z = \underset{v \in L}{\operatorname{argmax}} \sum_{y \in N} I(v = \text{class}(c_y))$；
I(.)是一个指标函数，当其值为 true 时返回值为 1，否则返回 0。

8.2.2　若干议题

kNN 算法的性能会受几个关键因素的影响，其中之一便是 k 值的选择，如图 8.1 所示，有一个未标记的测试对象 x 和一些训练对象（分属于“+”或“−”类）。如果 k 值选得过小，那么结果就会对噪声点的影响特别敏感；反之，如果 k 值选得过大，在近邻中就可能包含太多别的类的点。最佳 k 值的估计可以借助于交叉验证方法。然而，需要指出的是，取 k=1 常常会得到比其他值好的结果，特别是在习题和研究中遇到的那些小数据集这个现象更显著。然而，要注意，在样本非常充足的情况下，选择较大的 k 值则能提高抗噪能力。

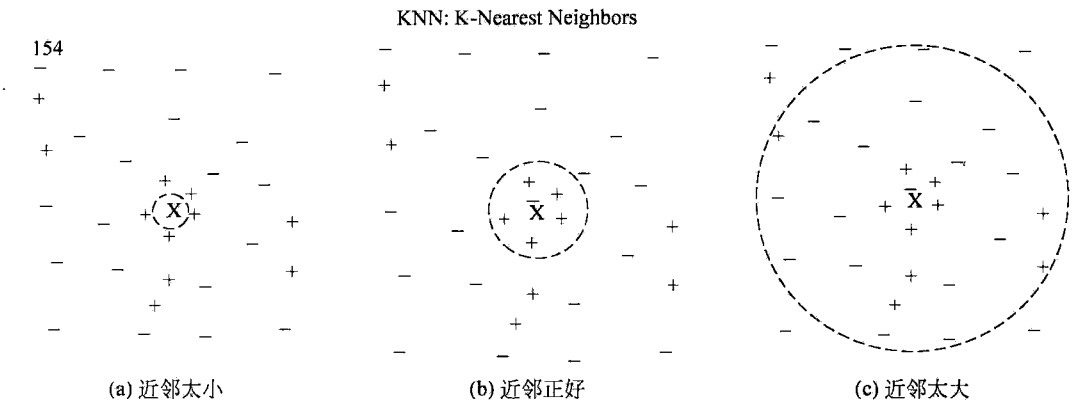
另一个问题是类别标签的综合，最简单的做法是采取投票，但是，如果不同近邻对象与测试对象间的距离差异很大的话，那么距离近的那些对象的类别对于判定测试对象所属的类作用应该更大。所以，一个稍复杂的方法是对每个投票依据距离进行加权，这种方法的一个附带的好处是 k 值的选择变得相对比较不敏感。加权的方法可以有多种，例如常常用距离平方的倒数作为权重因子：$w_i = 1/d(y,z)^2$。这相当于用下面式(8.1)替换了算法 8.1 的最后一步：

$$\text{基于距离权重的投票：} C_z = \underset{v \in L}{\operatorname{argmax}} \sum_{y \in N} w_i * I(v = \text{class}(c_y)) \tag{8.1}$$

距离测量的选择也是 kNN 算法的一个重要因素。一般情况下，常会使用欧几里德或曼哈顿距离[21]。对于给定的具有 n 个属性的两点 x 和 y，欧几里德距离和曼哈顿距离分别由下列公式来计算：

$$d(x,y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2} \tag{8.2}$$

$$d(x,y) = \sqrt{\sum_{k=1}^{n} |x_k - y_k|} \tag{8.3}$$



(a) 近邻太小　　(b) 近邻正好　　(c) 近邻太大

图 8.1　分别选用小、中、大的 k 值进行 k-最近邻分类

其中，x_k 和 y_k 分别是 x 和 y 第 k 个属性(部件)。

原则上各种测量方法都可以用于计算两点之间的距离，但是从概念上讲，最理想的距离测量方法应该具备这样的性质：对象之间的距离越小，它们属于同一类别的可能性越大。举例来说，如果将 kNN 算法应用于文本分类时，选用余弦距离就比欧氏距离更为合适。还有，通过为具有类别型属性值或者"类别-数值型混合"属性值的对象定义一个合适的距离测量方法，kNN 算法就可以用于挖掘这类数据。

有些距离测量方法会受到数据维数的影响，特别是欧几里德距离在属性数增加时判别能力会减弱。此外，可能需要对属性进行缩放，以防止距离测量结果被单个属性所主导。例如，在一个数据集中，人的身高区间是 1.5～1.8 米；体重区间是 90～300 磅；收入区间是 1 万～100 万元。如果距离测量没有对属性进行缩放，则收入将会主导距离计算和最终的分类结果。

8.2.3　软件实现

算法 8.1 非常简单，几乎用任何编程语言都很容易实现。但是，在这里我们还是要简单介绍几种已可直接使用的实现版本和一些变体，这样便于大家直接上手。最容易获得的一个可用的 kNN 算法的实现是 Weka[26]，该系统的一个名为 IBk 的函数实现了上面的算法 8.1，不仅如此，IBk 还允许用户从多种距离加权方法里进行选择，并且提供了一个选项以借助交叉验证来自动确定 k 值。

另一种非常流行的 kNN 实现的 PEBLS(基于实例的并行学习系统——Parallel Exemplar-Based Learning System)[5,19]，他属于 CMU 人工智能仓库[20]。从其网站上你可以了解到，"PEBLS 是一个最近邻学习系统，主要是为那些具有符号特征数据的应用设计的。"

8.3　示　　例

在本节中我们提供了两个运用 kNN 算法的例子。在这些例子中我们使用的都是 8.2 节中讲到的 Weka 包，具体使用的版本是 3.5.6。

首先，我们应用 kNN 算法对 Iris 数据集进行处理，该数据集可以从 UCI 机器学习资料库[2]中找到，当然它现在已经直接作为 Weka 的样本数据文件了。该数据集包含了 150 条关于花的数据，这些数据被等分成三类 Iris 物种：Setosa、Versicolor 和 Virginica。每朵花的数据所描述四项特征：花瓣长度、花瓣宽度、萼片长度和萼片宽度。

我们使用 IB1 算法来分类 Iris 数据集，IB1 是指 IBk 算法中的 k＝1 的情况，也就是说该算法仅使用最近的那一个邻居，在计算距离时用的是式(8.2)的欧氏距离。分类结果非常好，读者可以查验表 8.1 给出的混淆矩阵。

表 8.1　使用 Weka 的 kNN 分类器 IB1 在 Iris 数据集上得到的混淆矩阵

实际/预测	Setosa	Versicolor	Virginica
Setosa	50	0	0
Versicolor	0	47	3
Virginica	0	4	46

通过进一步的调查，我们可以发现其实这是一个很容易进行分类的数据集，因为不同物种的数据点在空间中相互间很好地分离开了。为了说明这一点，我们给出了在图 8.2 中所示的实际测量的花瓣长度和花瓣宽度中的数据图。某些 Versicolor 和 Virginica 在花瓣的长度和宽度方面相近从而会有一些混淆，但绝大部分情况下各物种间是很好区分的。虽然还有萼片宽度和萼片长度这两个变量，但他们增加的信息对判别性能而言几乎可以忽略不计。所以，这个最基本的 kNN 算法(IB1)取得的性能几乎是最好的，完全可以和各种 kNN 算法变体或者任何其他方法相当。

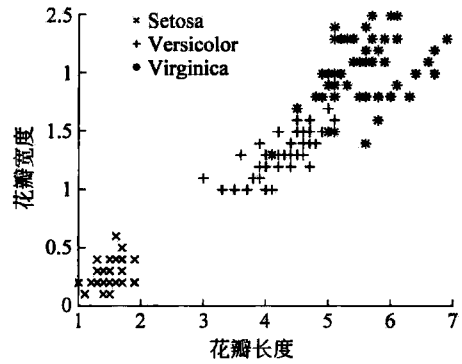


图 8.2　Iris 数据集使用花瓣长度和宽度进行分类的图示

第二个例子使用的 Ionosphere 数据也来自于 UCI。在这个数据集的数据对象描述了：发送到电离层的雷达信号和该信号是否带回了电离层的结构信息。其中，描述信号的属性有 34 个，描述类别的属性有 1 个。运用 IB1 算法，在原始数据集上得到了 86.3%准确度(基于 10 重交叉验证的评估)，而仅用前 9 个属性的情况下却得到了 89.4%准确度。也就是说，使用更少属性反而得到了更好的效果。再进一步，利用交叉验证选择的 2-最近邻(k＝2)的分类器可以获得 90.8%的精度。这些例子的混合矩阵分别如下面的表 8.2～表 8.4 所示。实验还发现，在本例中如果加大近处实例的权重则会导致精度的小幅下降，实际上对

精度的最大提升基本是由于减少属性的数量获得的。

表 8.2 使用 Weka 的 kNN 分类器 IB1 在原始 Ionosphere 数据集上得到的混淆矩阵

实际/预测	好信号	坏信号
好信号	85	41
坏信号	7	218

表 8.3 使用 Weka 的 kNN 分类器 IB1 在 Ionosphere 数据集上仅使用前 9 个属性得到的混淆矩阵

实际/预测	好信号	坏信号
好信号	100	26
坏信号	11	214

表 8.4 使用 Weka 的 kNN 分类器 IBk 在 Ionosphere 数据集上仅使用前 9 个属性在 k=2 时得到的混淆矩阵

实际/预测	好信号	坏信号
好信号	103	9
坏信号	23	216

8.4 高级主题

为了解决距离函数有关的问题，研究人员提出了一些方案，比如计算每一个属性的权重，或者面向特定数据集寻找与之适应的更有效的距离度量方法[13,15]。此外，人们还想到对训练对象本身进行加权，这样就可以实现加强高可信训练对象的作用，同时降低不可靠训练对象的影响。由 Cost 和 Salzberg 开发的 PEBLS 系统[5]就是使用这种方法的一个典型例子。

如前所述，kNN 分类器是一种惰性学习机，它不构建明确的模型，这与那些积极学习方法（例如，决策树，支持向量机等）是不同的。这样，虽然省去了构建模型的时间，但对未知对象的分类却更耗费时间，因为它需要通过计算来确定被预测对象的 k 个最近邻。所以，经典的 kNN 算法是需要计算在训练集中每个已标记对象同待预测对象之间的距离，这对有着大量训练对象的数据集来说计算成本尤其高昂。对此，人们已经发展了诸如多维访问方法（multidimensional access methods）[12]或快速逼近相似性搜索（fast approximate sinilarity search）[16]等技术，实现计算 k 最近邻距离的高效计算，其原理主要是充分利用数据本身的结构来避免对训练集中每个对象到测试对象间的距离进行计算。这些技术特别适用于低维数据，它可以有效降低计算成本同时又不影响分类的精度。Weka 的 IBk 例程也提供了一些多维访问方法（参见练习 4）。

最基本的 kNN 算法及它的一些变体，诸如属性加权和给对象加权等都是非常知名的。但是，还有一些相对而言不太为人所知的 kNN 技术改进。例如，在保持 kNN 算法分类精度的前提下，大幅度消解需要存储的数据通常是可以做到的，这种被称为“凝缩”的技术可以极大地加速对新对象[14]分类的速度。此外，还可以通过删除训练对象来提高分类的准确

性，这个办法被称为“编辑”[25]。针对 kNN 算法，研究人员基于邻近图（包括最近邻图、最小生成树、相对近邻图、Delaunay 三角化以及 Gabriel 图等）做了大量的研究工作。Toussaint 在论文[22~24]中强调了邻近图的观点，对相关工作进行了一个综述并指出了一些有待研究的开放性问题。

此外，还有其他一些重要的资源，包括 Dasarathy 的论文集[7]，Devroye、Gyorfi 和 Lugosi[8]写的书，Bezdek 提出并研究了一种模糊 kNN 算法[4]，最后，在线资源 Annotated Computer Vision Bibliography[18]为 kNN 算法提供了非常丰富的参考书目。

8.5 习　　题

1. 请下载数据挖掘软件 Weka 和 UCI 数据挖掘资料库中的数据集 Iris，然后重复本章性能分析相关的实验。

2. 请证明，在合理条件下，最近邻规则错误是贝叶斯错误的 2 倍。

3. 请证明，广义 kNN 的错误可以渐进达到贝叶斯错误，所以可以用作贝叶斯错误的近似。

4. 各种空间方法或多维方法可以用来加速最近邻算法，例如 k-d 树就是这样一种方法，请估计该方法能多大程度上节省时间。评述：Weka 的 IBk 算法可以指定返现最近邻的方法，可以选更大的 UCI 数据集进行测试，例如在 abalone 数据集上作性别预测。

5. 考虑表 8.5 所示的一维数据集。

表 8.5　习题 5 所用数据集

x	1.5	2.5	3.5	4.5	5.0	5.5	5.75	6.5	7.5	10.5
y	+	+	−	−	−	+	+	−	+	+

(a) 给定表 8.5 所示的数据点，请根据 x=5.5 的 1−、3−、6− 和 9 最近邻计算其所属类别。

(b) 用式(8.1)的加权 kNN 重复以上实验。

6. 请评述一下在 kNN 中用 consine 度量文本相似度而非文本距离的方法。

7. 请论述核密度估计及其与 kNN 的关系。

8. 如果用户不但希望对未知实例进行分类，还想知道为什么这么分？此时，应该选择哪种分类器：决策树还是 kNN？

9. 在很多数据分析任务中都用采样来减少数据点的数量，请评述一下采样用于 kNN。

10. 请讨论一下当每个实例有多个类别标签和/或类别成层次化体系时如何用 kNN 进行分类。

致　　谢

本工作受到以下项目的资助 NSF Grant CNS-0551551，NSFITR GrantACI-0325949，NSF Grant IIS-0308264 和 NSF Grant IIS-0713227，明尼苏达大学数字技术中心和超算研

究所提供了计算工具。

参考文献

[1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Mach. Learn.*, 6(1): 37-66, January 1991.

[2] A. Asuncion and D. Newman. UCI Machine Learning Repository, 2007.

[3] B. Bartsch-Spörl, M. Lenz, and A. Hübner. Case-based reasoning-survey and future directions. In F. Puppe, editor, *XPS-99: Knowledge-Based Systems-Survey and Future Directions, 5th Biannual German Conference on Knowledge-Based Systems*, volume 1570 of *Lecture Notes in Computer Science*, Wurzburg, Germany, March 3-5 1999. Springer.

[4] J. C. Bezdek, S. K. Chuah, and D. Leep. Generalized k-nearest neighbor rules. *Fuzzy Sets Syst.*, 18(3): 237-256, 1986.

[5] S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Mach. Learn.*, 10(1): 57-78, 1993.

[6] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1): 21-27, January 1967.

[7] B. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. Computer Society Press, 1991.

[8] L. Devroye, L. Gyorfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, 1996.

[9] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, November 2000.

[10] E. Fix and J. J. Hodges. Discriminatory analysis: Non-parametric discrimination: Consistency properties. Technical report, USAF School of Aviation Medicine, 1951.

[11] E. Fix and J. J. Hodges. Discriminatory analysis: Non-parametric discrimination: Small sample performance. Technical report, USAF School of Aviation Medicine, 1952.

[12] V. Gaede and O. Günther. Multidimensional access methods. *ACM Comput. Surv.*, 30(2): 170-231, 1998.

[13] E.-H. Han, G. Karypis, andV. Kumar. Text categorization using weight adjusted k-nearest neighbor classification. In *PAKDD'01: Proceedings of the 5thPacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 53-65, London, UK, 2001. Springer-Verlag.

[14] P. Hart. The condensed nearest neighbor rule. *IEEE Trans. Inform.*, 14(5):515 516, May 1968.

[15] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(6): 607-616, 1996.

[16] M. E. Houle and J. Sakuma. Fast approximate similarity search in extremely high-dimensional data sets. In *ICDE'05: Proceedings of the 21st International Conference on Data Engineering*, pages 619-630, Washington, DC, 2005. IEEE Computer Society.

[17] M. Kuramochi and G. Karypis. Gene classification using expression profiles: A feasibility study. In *BIBE'01: Proceedings of the 2nd IEEE International Symposium on Bioinformatics and Bioengineering*, page 191,Washington, DC, 2001. IEEE Computer Society.

[18] K. Price. Nearest neighbor classification bibliography. http://www. visionbib. com/bibliography/

pattern621.html，2008. Part of the Annotated Computer Vision Bibliography.

[19] J. Rachlin，S. Kasif，S. Salzberg，and D. W. Aha. Towards a better understanding of memory-based reasoning systems. In *International Conference on Machine Learning*，pages 242-250，1994.

[20] S. Salzberg. PEBLS：Parallel exemplar-based learning system. http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/learning/systems/pebls/0.html，1994.

[21] P.-N. Tan，M. Steinbach，andV. Kumar. *Introduction to Data Minining*. Pearson Addison-Wesley，2006.

[22] G. T. Toussaint. Proximity graphs for nearest neighbor decision rules：Recent progress. In *Interface-2002*，*34th Symposium on Computing and Statistics*，Montreal，Canada，April 17-20，2002.

[23] G. T. Toussaint. Open problems in geometric methods for instance-based learning. In *Discrete and Computational Geometry*，volume 2866 of *Lecture Notes in Computer Science*，pages 273-283，December 6-9，2003.

[24] G. T. Toussaint. Geometric proximity graphs for improving nearest neighbor methods in instance-based learning and data mining. *Int. J. Comput. Geometry Appl.*，15(2)：101-150，2005.

[25] D. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Trans. Syst.，Man，and Cybernetics*，2：408-421，1972.

[26] I. H. Witten and E. Frank. *Data Mining：Practical Machine Learning Tools and Techniques，Second Edition*（*Morgan Kaufmann Series in Data Management Systems*）. Morgan Kaufmann，June 2005.

第 9 章　Naive Bayes

David J. Hand

9.1　引言
9.2　算法描述
9.3　独立给力
9.4　模型扩展
9.5　软件实现
9.6　示例
　　9.6.1　例 1
　　9.6.2　例 2
9.7　高级主题
9.8　习题
参考文献

9.1 引 言

给定一个对象的集合，每个对象用一个(已知的)向量表示并且属于一个(已知的)类别，我们的目标是：构造出一个规则，使得对于未曾见过的一个对象(仅有已知的向量，不知其所属类别)，该规则能为其确定类别。这类被称为“有监督分类”的问题非常普遍，相应的规则构造方法也已经很成熟了。其中一个非常重要的方法就是朴素贝叶斯(Naive Bayes)，有时也被称为傻瓜贝叶斯、朴素贝叶斯或独立贝叶斯。这个方法非常受重视，这是因为：首先，它很容易构造，模型参数的估计不需要任何复杂的迭代求解框架，因此该方法非常适用于规模巨大的数据集；第二，它很容易解释，因此即便是不熟悉分类技术的用户也能够理解此方法是如何运作的；第三，更重要的是，它的分类效果往往非常好，对于任何应用，朴素贝叶斯即便不是最好的分类方法，通常也是非常稳健的。例如，早前，在一个比较监督分类方法的经典研究中，Titterington 等人(1981)发现这个独立性模型产生了最好的整体效果，Mani 等人(1997)发现了这个独立性模型对预测乳腺癌复发极为有效。此后，Hand 和 Yu (2001)基于很多案例报告了朴素贝叶斯方法令人惊讶的良好效果，Domingos 和 Pazzani (1997)在一些深入的实证比较中也得到了相同的结论。当然，也有其他一些研究表明这个方法的性能相对较低，关于此类研究的评述可以参考 Jamain 和 Hand(2008)。

为了方便，本章描述的大部分案例只涉及两个类别。实际上，在很多时候最重要的情形是自然分成两类的(对/错，是/否，好/坏，隐/现等)。但是，朴素贝叶斯本身是允许类别数多于两个的。

将类别记为 i=0,1，我们的目标是使用类别已知的初始对象(即训练数据)来构造一个打分器，使得获得较大分值的对象同类别 1 关联而获得较小分值的对象同类别 0 关联。打分器对新对象会给出其分值，将该对象的得分同某个预定的“分类阈值”进行比较即可实现分类，得分大于阈值就分到类别 1，小于阈值就分到类别 0。

有监督分类任务有两种基本范型，分别是诊断型和采样型。诊断型的关注点集中于两个类别之间的不同——即对两个类别进行判别。而采样型的关注点集中于两类个体分布的不同，通过对个体分布的比较间接实现对类别的比较。当然，我们还可以从其他角度来观察朴素贝叶斯。

9.2 算法描述

此处我们基于采样范型来阐述朴素贝叶斯模型，首先给出一些量的定义：$P(i|x)$表示一个测量向量为 $x=(x_1, \cdots, x_p)$的对象属于类别 i 的概率；$f(x|i)$表示 x 关于类别 i 的条件分布；$P(i)$为不知道对象自身任何信息的情况下该对象属于类别 i 的概率(即类别 i 的先验概率)，$f(x)$为两个类别的总的混合分布：

$$f(x) = f(x \mid 0)P(0) + f(x \mid 1)P(1)$$

很明显，如果对 $P(i|x)$的估计能得到一个合适的分数，可以将之用于分类规则，此外，我们还需要一个合适的分类阈值，这样就可以完成分类了。例如，最常使用的阈值是 1/2，

为了处理一些不同类型的分类错误，有时还需使用一些更加复杂的阈值选择方法。

根据贝叶斯定理，立即可以得到 $P(i|x)=f(x|i)P(i)/f(x)$，我们只要能估计出每个 $P(i)$ 和每个 $f(x|i)$ 的值，就可以得到 $P(i|x)$ 的估计。

如果训练集是从总体分布 $f(x)$ 中抽取的简单随机样本，那么根据属于类别 i 的对象在训练集里的比例就能直接估计出 $P(i)$。但有时候，获得训练集的方法比较复杂，例如某些问题中类别不均衡，一个类别比另一个大很多（在信用卡欺诈检查任务中，只有千分之一的交易存在欺诈；在罕见疾病检查等任务中，该比例甚至更极端）。这种情况下，通常要对类别进行欠采样，比如仅将大类的十分之一或者百分之一的可用数据放到训练集里。这样就有必要对直接观测到的训练集里类别的比例进行加权，来校正 $P(i)$ 的估计值。通常，如果不是用简单随机样本获得观测，就需要仔细研究如何获得 $P(i)$ 的最佳估计。

朴素贝叶斯方法的核心在于对 $f(x|i)$ 的估计。朴素贝叶斯方法假设对于每个类别，x 的分量都是相互独立的，所以有 $f(x|i)=\prod_{j=1}^{p} f(x_j|i)$，正是这个原因该方法也被称为“独立贝叶斯”。那么对 $f(x|i)$ 的估计也就仅需估计每个单变量的边际分布 $f(x_j|i)$，$(j=1,\cdots,p;\ i=0,1)$。这样，p 维多变量估计问题就被约减为 p 个单变量估计问题。相对于多变量分布，单变量分布估计更简单也被研究得更透彻，达到同一估计精度需要的训练集规模更小。

如果边际分布 $f(x_j|i)$ 是离散的且 x_j 仅含有少量取值，可以用简单的多项式直方图估计每个 $f(x_j|i)$。这种方法很直接，也正是朴素贝叶斯最常用的估计算法，很多软件实现采用了这种方法处理离散变量。实际上，很多软件实现还直接将连续变量（年龄、体重和收入等）的定义域分割成小的单元，这样就可以将多项式直方图估计方法应用到所有变量上了。乍看这种策略可能很弱，因直方图相邻单元间的任何连续性都丧失了，同时，单元覆盖必须足够宽以包含足够多的数据点，这样才能得到精确的概率估计。但从另一方面看，这种方法可以作为任何单变量分布的通用非参数估计方法，所以避免了引入任何分布假设。因为这是一种非线性变换，所以不同 $f(x_j|i)$ 估计之间的关系不一定是关于 x_j 单调的。

如果我们愿意花费更大的计算代价（更具体点，抛弃以计算过程简单见长的直方图估计方法），完全可以拟合出比仅使用一元变量边际分布更有力的模型。例如，可以假设具有特定参数形式的分布（正态、对数正态等），进而可以用标准和常见的估计方法来估计参数，甚至使用核密度估计等复杂的非参数方法。虽然这些复杂方法计算起来没有直方图估计方法快，但对于现代计算机而言差别也不大。倾向于使用直方图方法的一个理由是我们往往需要将所有变量离散化，我们在后面还将讨论到这个问题。

处于朴素贝叶斯方法核心位置的独立性假设貌似有点过头，因为在绝大多数实际问题中不太可能存在完全的独立性（你只要想想从实际应用的真实数据中计算出的协方差矩阵是对角阵的可能性有多大就明白了，实际上这种可能性几乎没有）。人们往往先入为主地认为，这个方法的根本性假设都不对，那它的效果一定好不到哪里去。但事实却是，该方法在很多真实应用上表现卓越。我们将在后面讨论造成这种违反直觉结果的原因。

到目前为止，我们用采样范型来阐述朴素贝叶斯，讨论了类条件分布的估计，假设每个分布的变量之间是独立的。朴素贝叶斯方法的一个迷人之处还在于我们能找到另一种与基本模型等价的形式，那就是我们可以使用严格单调变换对 $P(i|x)$ 和分类阈值同时变换，而分类结果不变。令 T 是严格单调递增变换，则有：

$$P(i \mid x) > P(i \mid y) \Leftrightarrow T(P(i \mid x)) > T(P(i \mid y))$$

进而，$P(i|x)>t \Leftrightarrow T(P(i|x))>T(t)$。这意味着，如果 t 是 $P(i|x)$用以比较的分类阈值，那么将 $T(P(i|x))$与 $T(t)$来比较也会得到同样的分类结果(我们这里仅假设单调递增变换，其实对其扩展也很容易)。

如下的比率变换就是这样一种单调递增变换：

$$P(1 \mid x)/(1-P(1 \mid x)) = P(1 \mid x)/P(0 \mid x) \tag{9.1}$$

朴素贝叶斯方法假设在每个类中变量间是独立的，依据类别 i 的分布的形式是 $f(\mathbf{x} \mid i) = \prod_{j=1}^{p} f(x_j \mid i)$，可将比率 $P(1|x)/(1-P(1|x))$重写为：

$$\frac{P(1 \mid x)}{1-P(1 \mid x)} = \frac{P(1)\prod_{j=1}^{p} f(x_j \mid 1)}{P(0)\prod_{j=1}^{p} f(x_j \mid 0)} = \frac{P(1)}{P(0)}\prod_{j=1}^{p}\frac{f(x_j \mid 1)}{f(x_j \mid 0)} \tag{9.2}$$

另外，对数变换也是单调递增的(多个单调函数的组合仍然是单调的)，所以可以得到另外一个打分器是：

$$\ln\frac{P(1 \mid x)}{1-P(1 \mid x)} = \ln\frac{P(1)}{P(0)} + \sum_{j=1}^{p}\ln\frac{f(x_j \mid 1)}{f(x_j \mid 0)} \tag{9.3}$$

如果我们定义 $w_j(x_j)=\ln(f(x_j|1)/f(x_j|0))$和 $k=\ln\{P(1)/(P(0))\}$，就可以将式(9.3)化为分离变量的简单求和：

$$\ln\frac{P(1 \mid x)}{1-P(1 \mid x)} = k + \sum_{j=1}^{p} w_j(x_j) \tag{9.4}$$

分值 $S=k+\sum_{j=1}^{p} w_j(x_j)$ 是直接由 $P(1 \mid x)$ 的估计算出来的，这是一种诊断范型的形式。现在就可以很清楚地看到：朴素贝叶斯模型其实就是对原始 x_j 的变换的简单求和。

如果每个变量都是离散的或者是将连续变量分割为小的单元来离散化，式(9.4)的形式就特别简单。记变量 x_j 的取其第 k_j 个单元的值是 $x_j^{(k_j)}$，那么 $w_j(x_j^{(k_j)})$就是“两个份额”之比的对数，即：类别 1 的点在变量 x_j 上取值落入的第 k_j 个单元的份额比上类别 0 的点在变量 x_j 上取值落入的第 k_j 个单元的份额。有些应用中将这个 $w_j(x_j^{(k_j)})$称为证据权重，即第 j 个变量对总分值的贡献，或者是说第 j 个变量为将对象归于类别－1 提供的证据。根据这些证据权重，可以识别出哪些变量对于判断任一特定对象的类别归属是重要的(这在一些应用中极为重要，如在个人银行业务中的信用打分环节，法律规定如果拒绝放贷就必须给出理由)。

9.3　独 立 给 力

关于每个类中变量 x_j 的独立性假设暗示了朴素贝叶斯模型可能是过度受限的。毕竟在实际问题中极少有变量独立的情形。但是，反过来说正是由于各种因素混杂在一起，反倒使得该假设并不像看上去的那样不妥(Hand 和 Yu，2001)。

首先，p 个一元变量边际分布的复杂性远小于一个 p 元变量的联合分布。这意味着，为了达到同样的模型精度，独立模型比非独立模型需要更少的数据点。换句话说，如果将模型

限制为变量具有类内独立性，那么基于给定可用样本的估计的方差就比较小。当然，如果该假设与实际不符，就会有偏差增大的风险。这就是经典的“偏差-方差”均衡原理的体现，该原理并不限于朴素贝叶斯模型，它适用于所有数据分析模型。

为了减少独立性假设带来的偏差风险，人们提出了基本朴素贝叶斯模型的一个简单的修正。为了理解这个修正背后的机理，先来分析一种极端特殊的情形：所有变量具有相同的边际分布并且它们是完全相关的。这意味着，给定一个类，第 x_j 个变量取值为 r 的概率和其他所有变量是一样的，在这种完全相关的情况下，朴素贝叶斯的估计是：

$$\frac{P(1 \mid x)}{P(0 \mid x)} = \frac{P(1)}{P(0)}\left[\frac{f(x_k \mid 1)}{f(x_k \mid 0)}\right]^p$$

此处 $k \in \{1, \cdots, p\}$ 可以任意选择。此时，真实优势比(odds ratio)是：

$$\frac{P(1 \mid x)}{P(0 \mid x)} = \frac{P(1)}{P(0)}\frac{f(x_k \mid 1)}{f(x_k \mid 0)}$$

如果 $f(x_k|1)/f(x_k|0)$ 大于 1，相关性的存在就使得朴素贝叶斯估计倾向于高估 $P(1|x)/P(0|x)$；反之，如果 $f(x_k|1)/f(x_k|0)$ 小于 1，相关性的存在就使得朴素贝叶斯估计倾向于低估 $P(1|x)/P(0|x)$。从这个现象立刻就得出一个修改了朴素贝叶斯估计的策略，就是对 $P(1|x)/P(0|x)$ 取小于 1 的幂，将整体估计向真实优势比收缩。这就形成了一种改良的朴素贝叶斯估计：

$$\frac{P(1 \mid x)}{P(0 \mid x)} = \frac{f(x \mid 1)P(1)}{f(x \mid 0)P(0)} = \frac{P(1)}{P(0)}\prod_{j=1}^{p}\left[\frac{f(x_j \mid 1)}{f(x_j \mid 0)}\right]^{\beta}$$

此处 $\beta < 1$。通常要搜索 β 的所有可能取值，用交叉验证等方法对每个取值对应的模型进行评估，最后确定能产生最好预测结果的 β 值。如果用式(9.4)分析，我们就能看到这相当于给 $w_j(x_j)$ 增加一个收缩系数。

让独立性假设显得有些道理的第二个原因是，通常会先对数据执行一个变量选择过程，例如线性回归中的变量选择方法。该过程将高度相关变量剔除，这些变量对分类的贡献非常相似，剩下的变量之间的关系可能接近于独立了。

独立性假设破坏力不大的第三个原因是，对于分类任务而言决策面才是重要的。虽然独立性假设可能会导致差的概率估计或差的 $P(1|x)/P(0|x)$ 比率估计，但这并不意味着估计得到的决策面和真实的决策面就相差很远(或者不同)。比如考虑一种情形：两个服从多变量正态分布的类别，有相同的(非对角)协方差矩阵，它们均值的差值向量平行于协方差矩阵的第一主轴，那么对应的最优决策面是线性的，并且用真实协方差矩阵和用独立性假设得到的结果是一样的。

最后要说明的是，朴素贝叶斯模型所产生的决策面可以具有复杂的非线性形状：决策面关于 $w_j(x_j)$ 是线性的，但关于原始变量 x_j 是高度的非线性，所以它能够拟合出相当复杂的曲面。

9.4 模型扩展

Naive Bayes 模型通常是很有效的。它的突出优点是计算简单，特别是离散变量模型。这个模型容易理解和解释，式(9.4)的逐点打分的形式特别凸显了这个特色，这正是该模型

被广泛使用的重要原因。可是，它的简单性和独立性这一核心假设与诸多真实情况不符，所以研究人员提出很多扩展力图来提高它的预测准确性。

在前面我们看到，收缩概率估计可以抑制独立性假设。此外，收缩也可被用来改善过于粗糙的多项式估计。如果第 j 个离散预测变量 x_j 有 c_r 种取值，记 n_{jr} 为全部 n 个对象在该变量取第 r 种值的次数，那么一个新对象在该变量上取第 r 种值的概率本来应该是 n_{jr}/n（多项式估计），但在实际中通常用它的一个校正值 $(n_{jr}+c_r^{-1})/(n+1)$，这个方法有时也被称为拉普拉斯校正，可以用贝叶斯统计理论对此给出一个直接的解释。当可用的样本大小和设定的单元宽度使得在一个取值上只有少量的对象时，这种校正方法特别有用。

减弱独立性假设的最显然的途径是为 x 在每个类中的分布模型引入额外的表示变量间相互作用的数学项。很多改进方法遵循这种思路，但是要知道这么做必然会引入复杂性，牺牲了贝叶斯模型的简单和优雅。当 x 中两个变量的相互作用被引入到模型中时，就不能仅仅基于单变量边际分布进行估计了。

类别 i 上 x 的（任意）联合分布都可以展开为：

$$f(x \mid i) = f(x_1 \mid i) f(x_2 \mid x_1, i) f(x_3 \mid x_1, x_2, i), \cdots, f(x_p \mid x_1, x_2, \cdots, x_{p-1}, i) \quad (9.5)$$

简化式中的各个条件概率项就得到近似结果。如果对于所有的 j，都采取 $f(x_j \mid x_1, \cdots, x_{j-1}, i) = f(x_j \mid i)$ 这种极端的简化方式，得到的正是朴素贝叶斯方法。显然，处于这两个极端之间的模型也是可用的。如果所有变量都是离散的，可以用对数线性模型很好地建模变量间任意程度的相互作用，对于连续变量的情形，使用图模型和条件独立建模语言是比较合适的。一个常用的例子是马尔可夫模型，它有很多应用：

$$f(x \mid i) = f(x_1 \mid i) f(x_2 \mid x_1, i) f(x_3 \mid x_2, i), \cdots, f(x_p \mid x_{p-1}, i) \quad (9.6)$$

它等价于使用了二元边际分布的一个子集，而不像朴素贝叶斯模型仅使用单个变量的边际分布。

另一种扩展是将朴素贝叶斯模型和树方法进行结合（Langley，1993 等）。比如，根据对象在一部分变量上的取值将总体切分成若干子集，然后为每个子集拟合相应的朴素贝叶斯模型。这种模型在某些应用中很流行，它们也被称为分段打分卡。分段是实现变量相互作用的一种手段，它克服了单一的整体独立模型在这方面的困难。

将朴素贝叶斯模型嵌入到高阶模型中的另一种方法是使用多分类器系统，例如随机森林和推举方法。

朴素贝叶斯模型和另一种非常重要监督分类模型——逻辑斯蒂回归模型有非常密切的关系。这个模型出自统计社区，在医药、银行、市场等领域中得到非常广泛的使用。逻辑斯蒂回归模型比朴素贝叶斯模型更强大，但代价是需要更复杂的估计框架。虽然它也有和朴素贝叶斯模型相似的简单形式，但是参数估计（例如 $w_j(x_j^{(k_j)})$）不能简单地使用比例估计，必须使用迭代算法。

在上述对朴素贝叶斯模型的探讨中，我们是借助于独立性假设才得到分解式（9.2）的。如果我们用 $g(x)\prod_{j=1}^{p} h_1(x_j)$ 和 $g(x)\prod_{j=1}^{p} h_0(x_j)$ 分别对 $f(x \mid 1)$ 和 $f(x \mid 0)$ 进行建模，相应的比率形式的模型也可以得到这样的分解式（假设此处两个 g(x) 是一样的）。如果 g(x) 不能被分解成关于分量（每个分量对应一个变量 x_j）的乘积，那么 x_j 就不是独立的，在 g(x) 中可以蕴含任意复杂的依赖结构，唯一的限制就是这种依赖在两个类别中要一致，也就是说

g(x)必须是 f(x|1)和 f(x|0)的公因子。对 f(x|i)进行因式分解我们可以得到：

$$\frac{P(1\mid x)}{1-P(1\mid x)}=\frac{P(1)g(x)\prod_{j=1}^{p}h_1(x_j)}{P(0)g(x)\prod_{j=1}^{p}h_0(x_j)}=\frac{P(1)}{P(0)}\cdot\frac{\prod_{j=1}^{p}h_1(x_j)}{\prod_{j=1}^{p}h_0(x_j)} \tag{9.7}$$

这里看到 g(x)被消掉了。比较式(9.7)和式(9.2)，虽然 $h_i(x_j)$和 $f(x_j\mid i)$不相等(除非 $g(x)\equiv 1$)，但其结构是相同的。注意在这个分解中，并不要求 $h_i(x_j)$非得是概率密度函数，只要 $g(x)\prod_{j=1}^{p}h_i(x_j)$是密度函数即可。

式(9.7)给出的模型和朴素贝叶斯模型一样简单，它们具有完全相同的形式。我们如果对其进行对数运算，就能得到一个式(9.4)那样的逐点打分模型。但是式(9.7)给出的模型要比朴素贝叶斯模型更灵活，因为它没有假设每个类中的变量 x_j 都是独立的。虽然逻辑斯蒂回归模型和朴素贝叶斯模型相比有同样的形式(当然参数值是不同的)并且更加灵活，但是它不能直接对单变量边际分布进行估计，必须使用一个迭代估计过程。标准统计教科书(比如 Collett，1991)一般都会给出逻辑斯蒂回归模型的参数估计算法，通常是用迭代比例加权最小二乘法(iterative proportional weighted least squares)来最大化似然度。

原始变量 x_j 的离散化是朴素贝叶斯模型的基础，我们可以对此进行扩展。广义加法模型(Hastie and Tibshirani，1990)其实就是对原始变量 x_j 的变换再进行加性组合。

朴素贝叶斯模型的巨大吸引力在于它的简单、优美、稳定以及模型构建过程和分类过程的快捷性。它是最早出现的形式化分类算法之一，其形式极尽简单，却有着令人惊讶的良好分类效果。为了使朴素贝叶斯模型更为灵活，统计、数据挖掘、机器学习和模式识别等社区提出了大量修改，但人们也逐渐认识到这样的修改必定会导致模型复杂化，而损害了它原本具有的简单性。

9.5 软件实现

朴素贝叶斯算法非常简单，所以它有很多实现。在 Web 上也有不少免费版本，开源软件 Weka(http://www.cs.waikato.ac.nz/ml/weka/)也实现了朴素贝叶斯算法，并且可以使用正态分布、核估计对单个变量进行建模，或者将变量离散化成类别值。

贝叶斯(Bayesian)这个术语有多种解释，它最常用在“朴素贝叶斯分类器”(Naive Bayes classifier)中，但这个词语可能会造成一些误解，需引起我们注意。其实，关于贝叶斯这个术语，我们特别应该了解“贝叶斯网络”(Bayesian networks)这一通用模型，朴素贝叶斯模型是它的一个特例。贝叶斯网络可以对变量间的多种相互作用进行建模(朴素贝叶斯要求变量独立)。Jamain 和 Hand(2005)对各种容易和朴素贝叶斯模型混淆的概念进行了详细剖析。

9.6 示 例

9.6.1 例 1

本例使用表 9.1 中的人工数据集来阐述朴素贝叶斯方法的原理。此处的目标是用该训

练数据集构造对新用户的 D 属性进行预测的规则。被预测的变量 D 在表中的最后一列，表示银行放贷的默认值(1 是不放，0 是放贷)。表的 1～3 列是用于预测的变量，分别是：受雇时间 T，单位是年；贷款额度 S，单位是美元；申请人身份 H，取值为房主(1)、租户(2)、其他(3)。事实上，朴素贝叶斯方法是检测信用欺诈的常规做法，只不过通常此类应用中训练集包含数以十万计的账户并且有很多变量，而朴素贝叶斯方法主要是作为前面提到的分段打分方法的叶节点。

表 9.1　例 1 的数据

受雇时间 T	贷款额度 S	申请人 H	违约者 D
5	10 000	1	0
20	10 000	1	0
1	25 000	1	0
1	15 000	3	0
15	2000	2	0
6	12 000	1	0
1	5000	2	1
12	8000	2	1
3	10 000	1	1
1	5000	3	1

受雇时间 T 是连续变量，我们要为它在估计每个类别上的分布 $f(T|i)$，$i=0,1$。为此，我们可以使用核方法进行估计，也可以假设某种参数化形式(比如对数正态分布就很适合用于这个变量)进行估计。当然，我们还可以使用朴素贝叶斯方法，就是将该变量的值域分成若干单元，然后统计各类别实例落入每个单元的比例，即可得到每个单元的概率估计。本例中我们采用第三种方法，根据客户受雇达到十年还是超过十年将 T 分成两个单元。这样就得到如下的概率估计：

$$\hat{f}(T<10 \mid D=0)=4/6,\quad \hat{f}(T\geqslant 10 \mid D=0)=2/6$$

$$\hat{f}(T<10 \mid D=1)=3/4,\quad \hat{f}(T\geqslant 10 \mid D=1)=1/4$$

类似地，我们把贷款额度 S 也分成≤10 000 和>10 000 两个单元。这样就得到如下的概率估计：

$$\hat{f}(S\leqslant 10\,000 \mid D=0)=3/6,\quad \hat{f}(S>10\,000 \mid D=0)=3/6$$

$$\hat{f}(S\leqslant 10\,000 \mid D=1)=3/4,\quad \hat{f}(S>10\,000 \mid D=1)=1/4$$

申请人身份的概率估计如下：

$$\hat{f}(H=1 \mid D=0)=4/6,\quad \hat{f}(H=2 \mid D=0)=1/6$$

$$\hat{f}(H=3 \mid D=0)=1/6,\quad \hat{f}(H=1 \mid D=1)=1/4$$

$$\hat{f}(H=2 \mid D=1)=2/4,\quad \hat{f}(H=3 \mid D=1)=1/4$$

假设收到一份新的贷款申请，由申请人他或她（这么措辞是刻意的，因为法律禁止使用性别作为贷款决策的参考因素。）受雇不满十年（T＜10），期望贷款 10 000 美元（s≤10 000），其身份是房主（H＝1）。那么比率$\hat{P}(1|x)/\hat{P}(0|x)$的估是：

$$\frac{P(1\mid x)}{P(0\mid x)}=\frac{P(1)}{P(0)}\prod_{j=1}^{p}\frac{\hat{f}(x_j\mid 1)}{\hat{f}(x_j\mid 0)}=\frac{P(1)}{P(0)}\times\frac{\hat{f}(T\mid 1)\,\hat{f}(S\mid 1)\,\hat{f}(H\mid 1)}{\hat{f}(T\mid 0)\,\hat{f}(S\mid 0)\,\hat{f}(H\mid 0)}$$

$$=\frac{4/10}{6/10}\times\frac{3/4\times 3/4\times 1/4}{4/6\times 4/6\times 3/6\times 4/6}=0.422$$

可以算出 P(1|x)＝1－P(0|x)；P(1|x)＝0.296；P(0|x)＝0.703，如果分类阈值取 0.5［即 P(1|x)＞0.5 则客户被分为类别 1，否则被分为类别 0］，结果是这个客户可能属于类别 0——非违约者。也就是说这个客户可能是一个不错的贷款对象。

9.6.2 例 2

朴素贝叶斯方法的一个较新的重要应用领域是垃圾邮件过滤。垃圾邮件是指用户不需要但却不请自来的电子邮件，往往是进行某种直销，比如提供金融或其他方面的可疑机会，其中有些就是所谓的钓鱼。垃圾邮件背后的原理是，即使回应率很低仍然有利可图。原因主要有（a）发邮件的成本忽略不计；（b）想发多少都行。垃圾邮件被自动发送到数以百万计的邮箱地址，光一个人每天就可能收到几百封垃圾邮件。对付如此大量的垃圾邮件，就移动光标和按下“删除”按钮这些动作也要消耗大量的时间。所以研究人员开发垃圾邮件过滤器，检查传入的电子邮件，并将它们分为垃圾邮件或非垃圾邮件。被分类成垃圾的邮件可以被自动删除或被放到一个保留文件夹中留待以后检查，当然也可以采用任何其他适当方式处理。

朴素贝叶斯模型用于垃圾邮件过滤器的构造最早要追溯到 Sahami 等人的开创性工作（Sahami 等，1998）。最简单的垃圾邮件过滤器为电子邮件中的每个单词指定一个二值变量以表示该词是否出现在目标邮件中。Bayes 模型自然也允许用二进制变量表示一些句法特征出现与否，诸如：标点符号、货币单位（$，£，€等）、词组，个人邮件还是邮件列表等。此外，还有一些非二元变量用于预测，例如邮件的来源、邮件主题中非字母数字的字符所占百分比等。可以看出潜在的变量的数量是非常大的，所以通常必须进行特征选择（回想一下为什么朴素贝叶斯虽然有粗糙的独立性假设但仍然能表现优异）。

垃圾邮件过滤中一个重要的问题是错分代价的不平衡性，即将合法非垃圾邮件错判为垃圾邮件比将反向错判代价高得多。这种代价不均衡和两个类别大小的差异确定分类阈值都非常重要。在 Sahami 等人的实验中，用来与 P（是垃圾邮件｜X）比较的分类阈值取的是 0.999。

朴素贝叶斯模型一个优点是用于计数变量和用于二值变量一样简单，以上所述的多变量二值垃圾邮件过滤器很容易被扩展成变量值有更复杂分布的模型。在前面的例子中，我们已经提到过使用多项式模型，并将连续变量的值域划分成两个以上的单元（例 1 数据集里的申请人身份变量就被分成了三值）。一些实验表明在垃圾邮件过滤任务上，用多项式分布刻画邮件中单词出现频率要优于仅使用二值变量刻画单词是否出现。Metsis 等（Metsis

等，2006)对朴素贝叶斯模型的不同版本进行了比较分析，该实验在将朴素贝叶斯模型在一些真实的电子邮件数据集上进行了评估，并且采用了多种边际变量的处理方法。

9.7 高级主题

朴素贝叶斯模型的魅力在于它的极其简单、易于(单变量)估计和基于证据权重的意义直观。模型的简单性有益于其可靠性，它使得可以获得边际分布的稳健估计。如果边际分布是类别型的，那每个单元就需要包含足够的数据点来产生精确的估计。所以研究人员深入研究了优化的变量单元划分方法。一种与贝叶斯模型最相应的方法是，分别处理每个变量，划分得到的单元数相同(这样做一般要优于划分成等长单元的方法)。更为复杂的方法是根据每个类别在每个单元上的相对数量来划分单元。最后，在划分单元时可考虑对每个类别(或两个类别)的分布的整体拟合，但这样做其实就偏离了原本的简单边际方法。Hand 和 Adams(Hand and Adams 2000)对这方面研究的一些调查进行了总结。

在几乎所有的数据分析任务中，数据缺失是一个潜在的问题。不能处理缺失数据的分类方法是有缺陷的。如果数据缺失是随机发生的，朴素贝叶斯模型处理就没有任何困难，因为它能从观测数据可以简单地得到边际分布的有效估计。但是，如果数据缺失是信息型的缺失，那处理过程就会非常复杂，这是一个值得进一步研究的领域。

现在越来越多的问题涉及动态数据和流数据集，朴素贝叶斯方法凭借其直接单估计方法的优势能很好地适应这些问题。

在生物信息学、基因组学、蛋白质学以及微阵列数据分析等某些领域中，被称为“小 n 大 P——样本少维数高”的问题是非常重要的。这些问题的特点是变量的数量比样本数量大得多。这就产生了奇异协方差矩阵、过拟合等非常困难的病态问题。为了克服这些问题，有必要引入一些假设或(等效地)以某种方式对估计进行收缩。在有监督分类任务中对付该问题一个可用的方法就是使用朴素贝叶斯模型。因为独立性这一内置的假设可以有效抵抗过拟合。这方面的改进思想越巧妙，得到的分类器就越复杂，所以应该注意寻求它们之间的最恰当的平衡。

9.8 习 题

1. 请使用某个开源软件包(比如 R)为两个类别中的每个类别生成 100 个实例的样本。其中，类别 1 服从：均值向量为零、协方差矩阵为单位矩阵的二元正态分布；类别 2 服从：均值向量为(0，2)、协方差矩阵为以(1，2)为对角线的对角矩阵的二元正态分布。请用这些数据拟合出一个 Naive Bayes 模型，假设其边际分布是正态分布，再画出决策面并确认它不是线性的。

2. 下表显示了从两个类别采样的二元分布，每个变量有 3 个取值。请确认每个类别中两个变量之间都是独立的。请以 1/2 为分类阈值来计算 Naive Bayes 分类器的决策面并确认它不是线性的。

144	144	144
144	144	144
144	144	144

9	90	9
90	900	90
9	90	9

3. 请用习题 2 中的数据为每个变量的每个取值计算证据权重，确认 Naive Bayes 分类器可以被表示成加权和的形式。

4. 下表显示了从两个类别采样的二元分布，每个变量有 3 个取值。请确认每个类别中两个变量之间不是独立的。请以 1/2 为分类阈值来计算 Naive Bayes 分类器的决策面并确认它不是最优的。

27	30	27
30	2700	30
27	30	27

432	48	432
48	432	48
432	48	432

5. 用多元正态分布生成的模拟数据，并不断提高变量间的相关性，请基于这些数据比较 Naive Bayes 分类器和某种简单线性分类规则的相对性能。

6. 请从 UCI 机器学习资料库中选取一个合适的数据集，将其连续变量分割成离散单元，再通过改变对应每个连续变量的离散单元的数量和宽度，来考察相应的效果。

7. 请基于习题 6 中的数据集来比较一下由 Naive Bayes 分类器和逻辑斯蒂回归生成的模型。

8. 一种常见的 Naive Bayes 分类器扩展方式是对数据进行分段，为每个数据片段单独构建一个 Naive Bayes 分类器。显然，如果能在此过程中为 Naive Bayes 分类器引入一些交互功能，将会显著提升其效能，请给出一些关于分段的指导性建议。

9. 本章讨论的基于独立性假设为每个类别建模一个分布的思想，很容易被推广到超过两个类别的情形。请给出多类别情形下具有证据权重形式的分类模型。

10. Naive Bayes 分类器的一个特别吸引人的地方是其估计算法非常简单，请给出新数据出现时对分类器进行顺序更新(在线学习)的规则。

参考文献

[1] Collett D. (1991) *Modelling Binary Data*. London: Chapman and Hall.

[2] Domingos P. and Pazzani M. (1997)Onthe optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29, 103-130.

[3] Hand D. J. and Adams N. M. (2000) Defining attributes for scorecard construction. *Journal of Applied Statistics*, 27, 527-540.

[4] Hand D. J. and Yu K. (2001) Idiot's Bayes-not so stupid after all? *International Statistical Review*, 69, 385-398.

[5] Hastie T. J. and Tibshirani R. J. (1990) *Generalized Additive Models*. London: Chapman and Hall.

[6] Jamain A. and Hand D. J. (2005) The naive Bayes mystery: A statistical detectivestory. *Pattern*

Recognition Letters, 26, 1752-1760.

[7] Jamain A. and Hand D. J. (2008) Mining supervised classification performance studies: A meta-analytic investigation. *Journal of Classification*, 25, 87-112.

[8] Langley P. (1993) Induction of recursive Bayesian classifiers. *Proceedings of the Eighth European Conference on Machine Learning*, Vienna, Austria: Springer-Verlag, 153-164.

[9] Mani S., Pazzani M. J., andWest J. (1997) Knowledge discovery from a breast cancerdatabase. *Lecture Notes in Artificial Intelligence*, 1211, 130-133.

[10] Metsis V., Androutsopoulos I., and Paliouras G. (2006) Spam filtering with naive Bayes-which naive Bayes? *CEAS 2006-Third Conference on Email and Anti-Spam*, Mountain View, California.

[11] Sahami M., Dumains S., Heckerman D., and Horvitz E. (1998) A Bayesian approach to filtering junk e-mail. In *Learning for Text Categorization-Papers from the AAAI Workshop*, Madison, Wisconsin, pp. 55 62.

[12] Titterington D. M., Murray G. D., Murray L. S., Spiegelhalter D. J., Skene A. M., Habbema J. D. F., and Gelpke G. J. (1981) Comparison of discrimination techniques applied to a complex data set of head injured patients. *Journal of the Royal Statistical Society*, *Series A*, 144, 145-175.

第 10 章　CART：分类和回归树

Dan Steinberg

10.1　前身
10.2　概述
10.3　示例
10.4　算法描述
10.5　分裂准则
10.6　先验概率和类别均衡
10.7　缺失值的处理
10.8　属性的重要度
10.9　动态特征构造
10.10　代价敏感学习
10.11　停止准则、剪枝、树序列和树选择
10.12　概率树
10.13　理论基础
10.14　CART 之后的相关研究
10.15　可用软件
10.16　习题
参考文献

本章主要向读者介绍学术专著 *CART*：*Classification and Regression Trees*，该著作是由 Loe Breiman、Jerome Friedman、Richard Olshen 和 Charles Stone(BFOS)这四位学者在 1984 年合作出版的。他们关于 CART 算法的杰出工作被认为是人工智能、机器学习、非参数统计和数据挖掘等学科发展史上的一个里程碑。几位学者在这部著作中阐述了他们对决策树理论进行的系统性研究：发展出多项关于决策树的新技术；展示了复杂的树形结构数据分析中的多种实景；对树的大样本理论给出了令人信服的论述等。正是这些奠基性的工作确立了这部著作的学术地位。根据文献索引工具 SCI 和 SSCI 的统计，该著作的被引用次数超过 3000 次，而 Google Scholar 统计得到的被引用次数已经超过 8450 次。CART 算法的应用面非常广，尤其是在信用风险、定向营销、市场经济建模、电子工程、质量控制、生物、化学和临床医学等研究领域。此外，在图像压缩领域，CART 算法中使用的树形向量量化技术也有深远的影响。本文意在通过介绍这部著作的主要内容，来简要地阐述 CART 算法的基本知识。当然，我们也非常鼓励读者去研读原书，以推敲算法之精微，品味思想之灵动，体验实际之妙用。

10.1　前　　身

CART 算法首次将决策树的研究置于坚实的概率论和统计学基础之上，其分析和表述都十分严谨。当然，在机器学习领域中，CART 并不是第一种被研究的决策树。CART 的前身可以追溯到 Morgan 和 Sonquist 在 1963 年发明的 AID 树 (automatic interaction detection tree)，该算法采用自动递归的方式挖掘数据中蕴含的关系，试图模拟实际的调查数据分析任务中典型的迭代钻取过程。AID 虽然是一种很有潜力的工具，但是她缺乏理论基础。这项 19 世纪 60 年代的工作广受质疑(Einhorn，1972；Doyle，1973)，主要因为 AID 算法会对训练数据产生严重的过拟合现象，以至于得出的结果偏差非常大，特别是在小样本情况下更为显著。在 1973 年前，很多统计学家们都认为关于树的研究已经再没有发展空间了，这些保守的智者们坚持认识到树缺乏理论基础，在实际应用中是不可靠的甚至是危险的。与此相反，也有一些学者没有放弃，仍然坚持树的研究路线进行探索。Cover 和 Hart (1967)深入研究了大样本条件下最近邻分类器(nearest neighbor，NN)的性质，Richard Olshen 和 Jerome Friedman 受到这一工作的启发而认识到树也有理论上的显著优点和巨大的研究价值。Olshen 推测，既然 NN 分类器可以达到错分的 Cover-Hart 误差界，那么一棵被适当构建的树也应能得到类似的结果，因为树的末端的节点可以看成动态构建的 NN 分类器。所以，Cover 和 Hart 关于 NN 分类器的研究激励了 Olshen 对树的渐进性进行探索。凑巧的是，Fridman 在研究基于树的快速最近邻识别问题时采用了一种递归划分机制，该机制后来被纳入 CART 算法中了。Friedman 先是将 CART 的一个雏形编入斯坦福线性加速器中心 SLAC(Stanford Linear Acceleator Center)1975 年的研讨文集中，随后他又以短文的形式公开发表。Friedman 和 Olshen 在 SLAC 的同一实验室中工作，Friedman 对 CART 算法的关键技术进行攻关，Olshen 则构建了相关的数学基础。同时，在 Los Angeles 实验室，另外两个人 Leo Breiman 和 Charles Stone 也在进行着这方面的研究(Breiman and Stone，1978)。这两个独立的研究小组在 1978 年走到一起，四位作者将他们的研究工作汇总到一起并撰写出了那本关于 CART 的经典著作。

10.2 概 述

CART 决策树包含的基本过程主要有分裂、剪枝和树选择等。分裂过程是一个二叉递归划分过程，预测属性和目标属性的类型既可以是连续型也可以是标称型。数据应以其原始形式进行处理，不需也不应对数据进行分级(binning)变换。数据分裂的过程从根节点开始，最早是根节点的数据分裂出两个孩子，进而每个孩子的数据再继续分裂出(两个)孙子，该过程一直进行直到没有数据可分。由于 CART 算法中没有停止准则，所以树会一直生长到最大尺寸。接下来是剪枝过程，CART 使用的是一种被称为代价复杂度剪枝(cost-complexity pruning)的新的剪枝方法，该方法从最大树开始，每次选择训练数据上对整体性能贡献最小的那个(当然也可以是多个)分裂作为下一个剪枝对象，如此直到只剩下根节点。这样 CART 就会产生一系列嵌套的剪枝树(而不是一棵剪枝树)，所以需要从中选出一棵作为最优的决策树。树选择时需要用一份单独的测试数据来评估每棵剪枝树的预测性能。这里要注意，在进行树选择时 CART 和 C4.5 使用的树预测性能评估方法完全不同：C4.5(以及其他一些经典统计学方法)使用训练数据而 CART 使用单独的测试数据(或者使用交叉验证策略)。

除了上面提到的分裂、剪枝和树选择这几个基本过程，CART 算法还包括(是可选的)一些别的内容：如自动类别均衡、自动缺失值处理、代价敏感学习、动态特征构建、概率树估计，以及属性的重要度排序等。CART 的作者们用交叉验证评估剪枝树序列中的每棵树的性能，这是一项开创性的工作，其难点在于进行不同折数(fold)的交叉验证时，这些树的终端节点数量可能不一致。虽然 BFOS 在 20 世纪的 70 年代就已经论述了这些主题，但是直到现在，他们的做法有些情况下仍然是领先的。20 世纪 90 年代发表的许多学术文献，核心内容其实是这本 1984 年的 CART 著作中相关部分的重新发现。我们将在本章接下来的部分中逐一讨论 CART 算法的重要特性。

10.3 示 例

为了能让复杂的 CART 算法变得具体可感，我们先用一个比较容易理解的真实例子(此处对原始数据做了一些删减)来阐述一些要点。在 20 世纪 90 年代的早期，作者曾经帮助一家电信公司对当时的移动电话市场进行调查。那时移动电话还属新鲜事物，我们需要做的是识别对移动电话价格敏感的主要用户群以及该群体的特性。所以，我们设计的一项市场调查活动，用以测试潜在购买者对移动电话套餐的反应(肯定/否定)。所有潜在客户看到的移动电话和相应服务都是完全一致的，唯一不同的是套餐价格，根据调查的设计，价格是随机变动的。

此次调查总共走访了 830 户，其中 126 户表示要订购这项手机服务。我们的核心目标就是最大程度地找出订购者和非订购者之间的差别。表 10.1 中列出这次调查获得数据的概要。这个数据集包含一组选定的属性，HANDPRIC 属性是指移动电话的价格，USEPRIC 属性是指每分钟的通话费，其他的属性的名字很直白地显示了自身的意义。

表 10.1　示例数据集的综合统计信息

属　性	N	N Missing	% Missing	N Distince	平均	最小	最大
AGE	813	18	2.2	9	5.059	1	9
CITY	830	0	0	5	1.769	1	5
HANDPRIC	830	0	0	4	145.3	60	235
MARITAL	822	9	1.1	3	1.9015	1	3
PAGER	825	6	0.72	2	0.076 364	0	1
RENTHOUS	830	0	0	3	1.7906	1	3
RESPONSE	830	0	0	2	0.1518	0	1
SEX	819	12	1.4	2	1.4432	1	2
TELEBILC	768	63	7.6	6	54.199	8	116
TRAVTIME	651	180	22	5	2.318	1	5
USEPRICE	830	0	0	4	11.151	10	30

我们将用该数据集构建一棵 CART 分类树，使用所有属性（除 RESPONSE）来预测属性 RESPONSE，其中 MARITAL 和 CITY 是标称属性。决策树的生长过程就是对训练数据的递归划分，每次划分都是根据某个分裂准则对某个目标节点进行分裂。图 10.1 演示了这个过程——从位于树顶端的根点开始分裂。

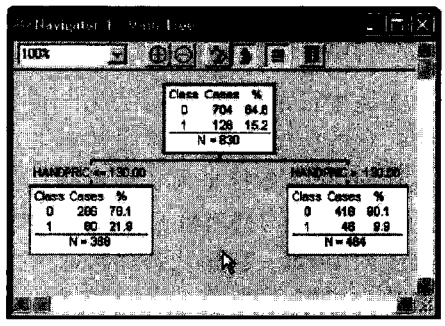


图 10.1　根节点的分裂

图中可见位于顶部的根节点包含了全部训练数据，由 704 个非客户（用 0 标示）和 126 个客户（用 1 标示）组成，共 830 个数据实例。其中，每一个实例都包含用于预测的 10 个属性，当然，存在一定程度的缺失值。CART 算法要通过分析训练数据来找到最佳的分裂器，期间测试所有的“属性-值”对儿评估分裂的性能。在图 10.1 中，我们看到分析的结果是：HANDPRIC 被选作最佳分裂器，其划分阈值是 130。所有 HANDPRIC 属性值小于或者等于 130 的实例都被分到了左边的子节点，其余实例的被分到右边。这次分裂产生了两个子数据集，他们的购买率（购买移动电话的人的比率）差异非常大：较低价格对应子集的购买率为 21.9%，另一个为 9.9%。很明显，从根节点上分裂器及其生成的两个子节点的差异看，预测性能还是很不错的。这里我们还可以注意到每次分裂都生成两个节点，即 CART 使用的是二叉分裂。

为了生成一棵完全树，CART 算法要重复上述的分裂过程：在已有分裂结果基础上，将根节点的两个子节点进行分裂即可生成根节点的孙子节点，再往下孙子节点还可以分裂成重孙节点，以此类推直到节点没有可分数据为止。在这个例子中，我们将会得到一棵包含了 81 个终端节点的“最大树”，这些位于树底端的终端节点不再继续分裂。

10.4 算法描述

如果要对 CART 算法进行全面阐述并理清所有技术细节，则需要很长的篇幅而且内容也会非常繁杂。首先，在树生长的过程中：分类和回归用到的分裂准则有很多种；连续型分裂器和类别型分裂器要用不同的方式处理，特别是类别型分裂器一般要被分为多个层级；提供对缺失值的处理等。在树生长完成之后就要进行树剪枝，而这是另一个很复杂的过程。最后，则是树的选择。在图 10.2 中，我们以伪代码的形式给出了一个简化了的树生长算法，该算法的完整的形式化描述可以参考那本 CART 专著。我们在这里仅提供一个非正式的高度简化版。

需要注意，这个简化算法仅仅是给出了一个构建最大可能（最大化）树的框架，而没有缺失值处理、类别赋值等部分的内容，也没有 CART 算法的其他核心细节。

```
开始：将所有数据分配给根节点并将根节点定义为叶节点
分裂：
New_split=0
For(树中每个叶节点)
    If(叶节点的样本数太少或者节点中所有点属于同一个类)
    goto GETNEXT
    根据分裂准则找到最优分类属性和阈值，将节点分裂为两个新的子节点
    New_split++;
GETNEXT:
```

图 10.2 简化的树生长算法

在树生长过程完成后，CART 算法接下来会生成一个嵌套的剪枝树序列。此处先给出一个简化的剪枝算法，该算法暂不考虑先验信息和算法代价，当然这和真正的 CART 算法是不同的，这么做仅是为了形式简洁和便于阅读。在该过程开始步骤中，以先前获得的最大树（T_{max}）为输入，将该树的那些在训练数据上不能提升预测精度的分裂剪掉。在该过程的剪枝步骤中，每一次迭代都移除树中当前最弱的链接，也就是移除在测试数据上性能提升最小的那些分裂。图 10.3 显示的就是这个简化算法，注意所有剪枝操作都被限定在两个叶节点的父节点上。

```
定义：r(t)=节点 t 中，训练数据被误分的比例
      p(t)=节点 t 中包含的训练实例数目
      R(t)=r(t) * p(t)
      t_left=节点 t 的左子树
      t_right=节点 t 的右子树
      |T|=树 T 中叶节点的总数
开始：Tmax=最大生长树
      Current_Tree=Tmax
      For(所有叶节点的父节点 t)
          将所有 R(t)=R(t_left)+R(t_right)的分裂点移除
      Current_tree=剪枝后的 Tmax
剪枝：If |Current_tree|=1 then goto DONE
      For(所有叶节点的父节点 t)
          将 R(t)-R(t_left)-R(t_right)最小的那个点移除
      Current_tree=剪枝后的 Current_Tree
```

图 10.3 简化的剪枝算法

而实际的 CART 算法和上面说到的简化算法有所不同，主要是用到了一种关于节点的惩罚机制，利用该机制可以在一次剪枝的动作中移除掉整棵子树。在那本专著中可以找到这个过程的清晰叙述。我们现在将针对 CART 算法的主要方面进行深入的讨论。

10.5　分裂准则

CART 算法的分裂规则通常采取以下的形式：

```
An instance goes left if CONDITION, and goes right otherwise
```

其中，CONDITION 是一个条件表达式。对于连续值属性，表达式就形如“(属性 Xi≤C)”；对于类别型或者标称型属性，表达式就是一种列表成员判定的运算。例如，基于变量 city 的分裂规则可写作：

```
An instance goes left if city is in {Chicago, Detroit, Nashville} and goes
right otherwise
```

在确定分裂准则后，CART 的最优分裂算法可以自动确定分裂器和分裂点。在未分组的(unbinned)原始数据上对属性实施保序变换(如取对数，取根号，取指数等操作)后，CART 算法的最优分裂是不变的。CART 的作者认为两路分裂比多路分裂更好，因为：(1)两路分裂比多路分裂导致数据碎片化的速度慢；(2)允许在一个属性上的重复分裂，这样可以在一个属性上产生足够多的分裂。两路分裂带来的树预测性能提升足以弥补其相应的树易读性损失。

在 CART 专著的作者讨论例子中，用到了分类树的四种分裂准则(Gini、Twoing、ordered Twoing、symmetric Gini)，但主要还是集中在 Gini 准则上。Gini 准则与非常著名的熵准则(即信息增益)很类似。

对于一个二元分类(0/1)的目标任务，一个节点 t 的“不纯度的 Gini 度量”公式为：

$$G(t)=1-p(t)^2-(1-p(t))^2$$

p(t)表示的是类别 1 在节点 t 上的相对频率(可能会加权)。G(t)的计算如果采用 $G(t)=-p(t)\ln p(t)-(1-p(t))\ln(1-p(t))$ 的话其实就是原先的熵准则。父节点 P 分裂出左子节点 L 和右子节点 R 后，获得的提升(增益)为：

$$I(P)=G(P)-qG(L)-(1-q)G(R)$$

这里的 q 是划分到左子节点的实例的比例(可能被加权)。相对于熵准则，CART 的作者们更偏爱 Gini 准则，因为后者较前者具有一些相对优势：她计算起来更快；可以很容易地嵌入对称代价(symmetrized costs)；能有效抑制“end cut”分裂——即生成一个很小(纯度相对较高)的孩子和一个很大的孩子(CART 软件的后续版本中也加入了熵准则，作为一种可选的分裂准则)。

Twoing 分裂准则针对目标属性，直接比较两个孩子节点中的类别分布：

$$I(\text{split}) = \left\{0.25(q(1-q))^u \sum_k \left| p_L(k) - p_R(k) \right| \right\}^2$$

k 表示是类别的索引，$p_L()$和 $p_R()$分别是左右孩子中目标类别的概率(这种分裂器是 Messenger 和 Mandell，1972 的一个修改版)。Twoing 准则对应的提升实质上是对左右子

节点的概率向量差异的度量。公式中前边的[0.25(q(1－q))]这项相当于对分裂生成的左右子节点大小不均衡的情况进行惩罚，显然当 q=0.5 时该项取最大值。指数项 u 是用户指定的，允许持续加强对左右子节点大小不均衡的惩罚。例如设置 u=10，类似于强制所有的分裂都产生在分裂属性的中值处。根据我们的实际经验，Twoing 准则在多分类任务和二分类任务中表现都很好。BFOS 还提出了一种 Twoing 分裂准则的变体，称为 ordered Twoing 分裂准则。此处的 ordered 指的是将所有类别进行排序。ordered Twoing 分裂准则本质上是用于分类任务的，但形式上她又具有一定程度的回归特性，每次分裂操作，她都是将低序号的类别同高序号的类别分开。

对于回归任务(连续型目标)，CART 提供了两种基本的分裂准则用于度量分裂的性能提升能力，一种是最小均方误差 LS(least squares，预测错误的平方和)准则，另一种是最小绝对偏差 LAD(least absolute deviation，预测错误的绝对值的和)准则。回归树和分类树类似，也要通过选择最优分裂来产生最大的性能提升。另外的三种分裂准则主要是用于代价敏感学习和概率树，所以我们在后面的相关部分再行讨论。

在我们移动电话的例子当中，我们首先计算根节点的 Gini 值，得到的根节点的不纯度 $1-(0.84819)^2-(0.15181)^2$；再分别计算每个子节点的 Gini 值；最后用根节点的 Gini 值减去左右子节点的 Gini 值(用节点的样本比重进行加权)，得到 0.00703 的性能提升(结果数值也许会受到与输入值和计算的精度的轻微影响)。用同样的方法 CART 算法可以算出其他可用属性的性能提升值(此处，我们在表 10.2 中列出了根节点的最优分裂属性 HANDPRIC 的前 5 个竞争者及其性能提升值)。

表 10.2 主分裂性能提升

	竞争者	分裂	性能提升
1	TELEBILC	50	0.006 883
2	USEPRICE	9.85	0.005 961
3	CITY	1,4,5	0.002 259
4	TRAVTIME	3.5	0.001 114
5	AGE	7.5	0.000 948

10.6 先验概率和类别均衡

确保类别均衡对机器学习在实际应用中非常重要的，如果训练集高度不均衡，很多的数据挖掘方法都不能得到好的效果。例如，对于大多数贷款机构，违约率一般低于所有用户的 5%；信用卡交易欺诈通常低于 1%；互联网广告的点击率通常远小于所有显示广告的 1%。常规做法是对训练数据集进行采样；使得最终生成的类别大小相近。显然，如果感兴趣的类别非常小，这种样本均衡方法获得的可用训练数据相对于原始的大训练集就很少。例如，在保险欺诈分析中，某公司识别出 70 个存在诈骗的案例，如果一定要分析者使用均衡的样本，那么就只能得到仅包含 140 个实例的样本了(70 个有诈骗的，70 个无诈骗的)。

幸运的是，CART 算法的作者在 1984 年就深入研究并明确解决了这个问题，从而使得

建模过程不再需要考虑任何样本均衡的问题。无论训练数据集有多失衡，CART 算法都可以将其自动消除，不需要建模人员采取其他操作、准备、采样或加权。总之，无需针对数据的均衡性进行任何预处理，可以直接拿来建模。

为了实现这一重要特性，CART 算法使用一种先验机制，其作用相当于对类别进行加权。这种机制是不可见的，因为 CART 算法报告的关于树的计数信息没有体现出先验。先验被嵌入到 CART 算法中判定分裂优劣的运算里了。在 CART 的默认的分类模式中，总是要计算每个节点关于根节点的类别频率的比值。这就相当于对数据自动重加权，对类别进行均衡。同时，也确保树的选择是遵循的最小化错误率原则也是建立在类别均衡的基础上。重加权隐藏在所有关于概率和提升的计算中，不需要用户干预。每个节点上样本包含的实例数量反应的是未加权的数据。对于一个二分类任务，节点 node 被分成类别 1 当且仅当：

$$\frac{N_1(\text{node})}{N_1(\text{root})} > \frac{N_0(\text{node})}{N_0(\text{root})}$$

我们观察这种计算方式可以发现，不管数据真实的类别分布如何，如果共有 K 个目标类别，就可以确保根节点中每个类别的概率都是 1/K。这种默认的模式在这部 CART 专著中被称为“先验相等”（“priors equal”）。她使得用户很容易处理任何失衡数据，而不需要做任何特别的均衡类别操作或者人为的加权方法构造等数据预处理。只要使用 CART 软件的默认设置就足以有效地处理失衡数据了，当然也可以将“priors data”选项关闭这种隐含的重加权功能。建模者也可以根据需要指定先验值集合，以反映学习代价或者反映训练数据和未来预测数据类别之间的差异。

注意：*先验设置和加权的不同之处在于先验不影响每个节点中的各类别样本的数量或者份额。先验影响的是每个节点的类别赋值和树生长过程中分裂的选择。*

分析人员通常依靠先验机制来实现类别均衡，但这并没有否定采样方法，即对不同类别进行不同比率的采样。相反，它让分析师在采样的时候有了更多更灵活的选择。

图 10.4 反映的是我们用先前的移动电话数据集生成的一个棵 CART 树，由于使用了“priors equal”设置，该树能很好地处理类别失衡（同意购买类别显著少于拒绝购买类别）的情况。

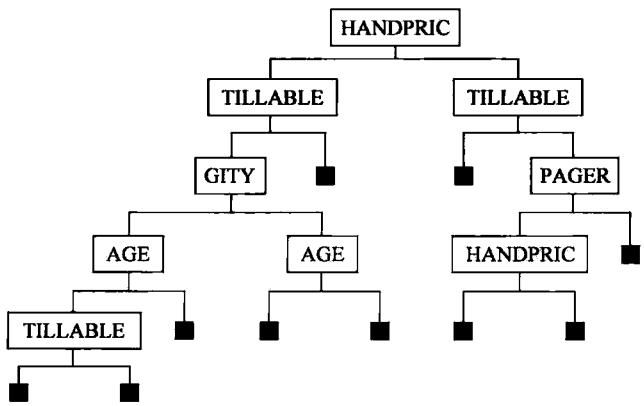


图 10.4　CART 树

CART 算法约定：基于连续型变量分裂时，把值大的实例分到右边子节点；基于标称型变量分列时，根据属性值列表中值的排列次序决定把对应实例分到左边还是右边。在图 10.4 中，叶节点用颜色来反应用户购买的相对概率，红色节点高于平均水平，蓝色节点是低于平均水平。虽然图 10.4 显示的只是全部信息的一小部分，但是足以告诉我们好消息：虽然移动电话这种新技术定价很高，但是那些使用“固定电话＋寻呼机”模式且通话量很大的客户还是非常倾向于使用移动电话这种新服务的。图 10.4 中也展示了 CART 算法如何多次使用同一个属性。看一下树的右半部分，分析那些使用固定电话（但没有寻呼机）的大通话量的客户的时候，我们看到又一次使用了 HANDPRIC 属性。在此处，我们知道这个客户群是愿意为此支付略高的价格，但不会出到最高价格（基于 HANDPRIC 属性的第二次分裂的阈值是 200）。

10.7 缺失值的处理

在现实世界中缺失值是频繁出现的，特别是商业数据库，这对所有建模者都是一个相当棘手但又必须处理的问题。CART 算法的主要贡献之一就是她采用完全自动的机制高效地处理缺失值的问题。决策树在 3 个层面上需要对缺失值进行处理：(a)分裂属性的评估；(b)训练数据跨过节点；(c)测试数据跨过节点并得到最终的类别赋值。对于(a)，在 CART 算法的第一个版中，严格要求分裂属性评估时只能使用在该属性上没有缺失值的那部分数据。在后来的版本中，CART 算法使用了一族惩罚机制来抑制提升值，从而反映出缺失值的影响（例如，如果一个变量在节点的 20％的记录上存在缺失值，那么他对该节点的提升值就会减少 20％或者 20％的一半等）。对于(b)和(c)，CART 算法的机制是为树的每个节点都找到“代理”或替代分裂器，无论在训练数据中是否有缺失值出现都会这样做。代理分裂器非常有用，她确保在无缺失训练数据上得到的树可以用来处理包含缺失值的新数据。这不仅与那些不从有缺失值训练数据学习的机器形成强烈的对比，同时也有别于那些必须从包含丢失值的训练集中学习缺失值处理的机器。Friedman(1975)建议把分裂属性上带有缺失值的实例同时分到左边和右边，而类别的赋值则依据所有包含该实例的节点加权平均。Quinlan 在研究缺失值处理时提出的新方法实际是 Friedman 方法的一个变体。我们评估 CART 算法中缺失值处理的代理分裂器策略时，发现效果非常好，可能 Quinlan 不知道他的实现其实是 Friedman 方法的一个近似。在文献（Friedman，Kohavi，Yun（1966））中，Friedman 指出了 CART 算法大概有 50％的代码是用于处理缺失值的，这不同于 Quinlan 的代理分裂机制的情形。

在 CART 算法中，缺失值的处理机制是完全自动的和节点局部自适应的，如表 10.3 所示。在树的每个节点上，分裂器都会导出节点数据的一个二元划分（x1≤c1 和 x1＞c1）。一个代理分裂器是能预测这种划分的单个属性，它也具有二元划分的形式（Z≤d 和 Z＞d）。换言之，每一个分裂器都是一个新的分类任务。

代理分裂器按关联分值进行排名，这个分值是代理分裂器相对于默认规则的增强，默认规则预测所有的实例都是分到一个大的节点中（这些都是在经过了 priors 调整后）。一个变

量必须超过默认规则的性能，才有资格作为代理（因此并不是总可以找到一个代理）。当在 CART 树中遇到缺失值的时候，这个实例划分到左边还是右边是决定于它排名最高的代理。如果这个代理的值也缺失了，那么就使用排名第二的代理，以此类推。如果所有的代理的值都丢失了，那么默认规则就是把所有实例都分到较大的那个子节点（这些都是在经过了 priors 调整后）中。

表 10.3　代理分裂器的性能提升

分裂器 TELEBILC 的提升为 0.023 722。

	代理	分裂	相关度	性能提升
1	MARITAL	1	0.14	0.001 864
2	TRAVTIME	2.5	0.11	0.006 068
3	AGE	3.5	0.09	0.000 412
4	CITY	2,3,5	0.07	0.004 229

再回到移动电话的例子，其中根节点的右子节点是通过 TELEBILC 属性来分裂的（固定电话的账单）。如果电话账单数据缺失了（例如这是一个新用户，电话公司关于他的信息非常少），不管这个实例是应该被分到左边还是右边，CART 算法都会去搜索所有的属性来找到一个最好的预测。

在这个例子里（见图 10.3），如果我们要预测固定电话花费是否够高（即大于 50）？那么，在所有可用的属性中，预测能力最强的属性是婚姻状况（没结过婚的人使用得少），然后依次是上班途中花费的时间、年龄，而最后是所在城市。代理相当于对分裂器进行解释，所以也可以将代理看作是分裂器的同义词。这里我们可以看到，电话费较低的人具有一些明显的倾向，如从没结过婚、生活接近与市中心、年轻，并集中在其中 3 个城市（本例子共涉及 5 个城市）。

10.8　属性的重要度

一个给定的属性可能会被决策树的多个节点选作分裂器，她在每个节点上都会一定程度地提升决策树的性能，提升的总量（要用各节点包含的训练数据的比例进行加权）就构成了该属性的重要度。这里也包括对代理属性（surrogates）的重要度的计算，即使一个变量从未用于节点分裂，仍有可能获得很高的重要度。这样，变量重要度的排名就能揭示属性间隐蔽的、非线性的关联。虽然可以仅计算分裂器的重要度，但是将分裂器的重要度和全部属性的重要度（分裂器属性和代理属性）统一排序是一个很有用的诊断方式。

在表 10.4 中属性 MARITAL、RENTHOUS、TRAVTIME 和 SEX 并没有被选作分裂器，但它们还是能在树中起到一定作用。这些属性被用作别的属性的代理，所以有明确的非 0 重要度。CART 算法的报告可以被配置成忽略代理的重要度，就如表 10.5 中展示的情况。

表 10.4 各种重要度(包括代理属性)

属性	得分	
TELEBILC	100.00							

10.10　代价敏感学习

代价的概念在统计决策理论中处于中心地位，但是代价敏感学习的研究受到普遍关注是在 Domingos(1999)之后。从那时起，针对代价敏感学习这个主题出现了许多专门的会议，同期也产生了大量这方面的研究论文。但是，我们看到早在这部 CART 专著就已经阐述了两种代价敏感学习的策略，而且用于描述 CART 算法的整个数学体系就是围绕着分类错误的代价展开的。可以定义一个实例被错误分类的代价为 C(i,j)，以表示实例的类别为 i 但被预测为类别 j 的情况。如无特别的声明，每个错误分类的代价取值为 1，当然对于所有类别 C(i,i)＝0。所有代价可用一个矩阵 C 来表示，每个类别都有对应的行和列。一棵分类树的总代价就是每个叶节点上的分类错误的代价的和。这种代价敏感学习机制的目的是要在树的生长和剪枝过程中考虑代价因素。

最容易想到的也是最直接的代价处理方法就是加权：对于每个类的所有实例都赋予一个相同的初始权重，当实例被误分时就提高该实例的权重，Ting(2002)重新发现了这种方法。在 CART 的实现中，加权是透明的，也就是说算法是按照最初未加权的形式报出所有节点的计数的。对于多类分类问题，BFOS 建议对错误分类代价矩阵按行进行汇总得到相对的类别权重，用作代价的近似。这种技巧不考虑代价矩阵内部的细节，但是因为非常简单而被大家广泛使用。对于 Gini 分裂准则，CART 的作者表示可以对代价矩阵进行对称化处理，这样就能将整个代价矩阵嵌入到分裂准则中去。基于这种“对称 Gini”(symGini)分裂准则的树生成方法对 C(i,j)和 C(i,k)之间的差异很敏感。所以，如果决策问题适合采用对称代价矩阵表示，那么用这种方法就会非常有效。与之相反，实例加权的方法则是对给定类 i 的所有错误分类赋予了相同的权重。BFOS 观察到使用全代价矩阵进行剪枝对于实现成本学习是非常必要的。

10.11　停止准则、剪枝、树序列和树选择

在最早期的决策树是不剪枝的，决策树会一直生长直到他们满足了一些停止条件，就得到最终结果树。在这部 CART 专著中，作者证明了没有一种停止树生长的准则可以保证识别出重要的数据结构(例如，二维异或问题)。因此，他们选择了不去停止树的生长，而是以足够大的结果树为原始素材，从中提取出最优的树。

剪枝机制严格地基于训练数据。首先，给出所谓的代价复杂度的定义：

$$Ra(T)=R(T)+a|T|$$

此处 R(T)表示树的训练样本的代价，|T|是树的叶节点数目，a 是施加在每个节点上惩罚因子。如果 a＝0，那么最小代价复杂度树就是最大树；如果 a 允许不断地增加，那么最小代价复杂度度树将会逐渐变小，因为那些树底部降低 R(T)幅度很小的分裂将会被砍掉。参数 a 从 0 开始一直增加，总可以达到一个值能将所有的分裂都剪掉。BFOS 证明了，在所有的有 Q 个叶节点的树中，用这种方法去提取出的树有最小的代价 R(Q)。这在实际中是非常重要的，他大幅度减少了搜索最优树的过程中需要测试的树的数目。剪枝过程包括两

步：首先，删除分裂(一个分裂产生两个叶节点)；然后，把两个叶节点吸收进他们的父节点，这样就可以用以一个节点代替两个叶节点了。用这种剪枝方法，从最大树中提取的子树的数量很大，具体要取决于该树的拓扑结构，但有时会超过|T|!/2。但是，基于代价复杂度剪枝，我们需要检验一个树的数目就要小很多。在我们的例子中，我们长出了一棵有 81 个叶节点的树，代价复杂度剪枝得到的子树序列仅含 28 棵子树，但是如果我们要去查找所有可能子树的话，我们必须去检查 25!=15 511 210 043 330 985 984 000 000 棵树。

所谓最优树，就是指在剪枝序列中的测试数据上代价最小的树。因为测试数据上的错误分类代价的度量受采样误差的影响，所以选择剪枝序列中哪棵树是最优树也具有一定程度的不确定性。事实上，错误曲线(分类错误率是树规模的函数)有一个有趣的特性，对于一个大的训练数据集，错误曲线在最小值附近趋于平坦。BFOS 推荐选择 1 标准差("1 SE")树，就是相对于最小代价的 0 标准差(0 SE)树有 1 个标准差代价的树。他们之所以青睐 1 标准差规则，是因为模拟表明：在各种情况下该规则生成的树的尺寸都很稳定，而 0 标准差(0 SE)规则生成的树的尺寸在不同情况下变化非常剧烈。

图 10.5 展示了一棵分类树剪枝过程中的一个中间状态，那些高亮的部分表示的是在下一步中要被代价复杂度剪枝算法移除掉的分裂。

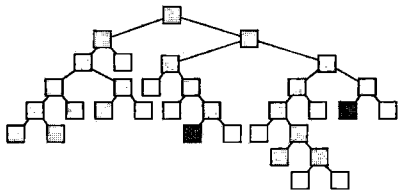


图 10.5 CART 剪枝过程中的一个中间状态

表 10.6 中的每一行对应一个被剪枝的子树，该剪枝过程始于一个具有 81 个叶节点的最大化树。剪枝序列覆盖了所有回到根的路径，因为我们必须允许我们的树没有对测试数据的预测能力。在测试数据上性能最好的树是有 40 个节点的 SE 0 树，与 SE 0 树有一个标准差的最小的树是 SE 1 树(有 35 个叶节点)。为简单起见，我们在先前的讨论中展示了有 10 个叶节点的次优子树的细节(图 10.4)。

表 10.6 CART 模型的全部树序列：全部嵌套子树

树	叶节点数	测试代价			训练代价	复杂度
1	81	0.635 461	+/−	0.046 451	0.197 939	0
2	78	0.646 239	+/−	0.046 608	0.200 442	0.000 438
3	71	0.640 309	+/−	0.046 406	0.210 385	0.000 72
4	67	0.638 889	+/−	0.046 395	0.217 487	0.000 898
5	66	0.632 373	+/−	0.046 249	0.219 494	0.001 013
6	61	0.635 214	+/−	0.046 271	0.231 94	0.001 255
7	57	0.643 151	+/−	0.046 427	0.242 131	0.001 284
8	50	0.639 475	+/−	0.046 303	0.262 017	0.001 43

续表

树	叶节点数	测试代价			训练代价	复杂度
9	42	0.592 442	+/−	0.044 947	0.289 254	0.001 709
10	40	0.584 506	+/−	0.044 696	0.296 356	0.001 786
11	35	0.611 156	+/−	0.045 432	0.317 663	0.002 141
12	32	0.633 049	+/−	0.045 407	0.331 868	0.002 377
13	31	0.635 891	+/−	0.045 425	0.336 963	0.002 558
14	30	0.638 731	+/−	0.045 442	0.342 307	0.002 682
15	29	0.674 738	+/−	0.046 296	0.347 989	0.002 851
16	25	0.677 918	+/−	0.045 841	0.374 143	0.003 279
17	24	0.659 204	+/−	0.045 366	0.381 245	0.003 561
18	17	0.648 764	+/−	0.044 401	0.431 548	0.003 603
19	16	0.692 798	+/−	0.044 574	0.442 911	0.005 692
20	15	0.725 379	+/−	0.045 85	0.455 695	0.006 402
21	13	0.756 539	+/−	0.046 819	0.486 269	0.007 653
22	10	0.785 534	+/−	0.046 752	0.539 75	0.008 924
23	9	0.784 542	+/−	0.045 015	0.563 898	0.012 084
24	7	0.784 542	+/−	0.045 015	0.620 536	0.014 169
25	6	0.784 542	+/−	0.045 015	0.650 253	0.014 868
26	4	0.784 542	+/−	0.045 015	0.710 43	0.015 054
27	2	0.907 265	+/−	0.047 939	0.771 329	0.015 235
28	1	1	+/−	0	1	0.114 345

10.12　概　率　树

近期有一系列的论文对概率树进行了非常深入地探讨，重点是研究概率树的性质和探索提高其性能的方法（参见 Provost and Domingos，2000）。其实，这部 CART 专著是最早对概率树进行详细讨论的文献，而 CART 软件也提供了专门用于生成“分类概率树”的分裂准则。分类树和概率树关键的不同点是：有的分裂会将生成的两个叶节点分配与他们父节点相同的类别，对于这种分裂概率树希望保留下来，而分类树不允许这样做（因为这样的分裂对最终的分类精度没有作用）。另外，概率树在剪枝方面也不同于分类树。因此，即便是基于完全相同的数据上，CART 算法建立的分类树和概率树的最终结构可能存在一些差异（当然这种差异一般并不显著）。概率树的主要缺点是，在叶节点中根据训练数据进行的概率估计是有偏的（对于二元分类任务，就是偏向类别 0 或偏向类别），而且叶节点的深度越深这种偏差越大。在最近机器学习文献中，有研究推荐使用拉普拉斯校正（Laplace

adjustment)去消减这种偏差(Provost and Domingos, 2002)。在这部 CART 专著中,提出了一种更加复杂的方法去调整叶子节点的概率估算方法——布雷曼校正"Breiman adjustment",不过在其他文献中很少提到这种方法。它上调了每个叶节点的错误分类率估计 r×(t),如下:

$$r\times(t)=r(t)+e/(q(t)+S)$$

此处的 r(t)是节点内训练样本的估计,q(t)是节点内训练样本的所占比例,待定参数 S 和 e 是给定树的训练错误率和测试错误率之差的函数。与拉普拉斯校正的方法不同,布雷曼校正不依赖于节点的初始预测概率,而且如果测试数据表明树没有过拟合的话,这重调整的幅度影响是非常小的。Bloch,Olshen 和 Walker(2002)详细讨论这个主题,通过一系列的实验,他们得出布雷曼校正的性能非常好。

10.13 理论基础

决策树方法被提出后逐渐显示出其有用性,但是早期的决策树研究工作可以说完全不涉及理论层面,而是在通过一些具体的例子,观察树的性能表现,进而分析和总结出树的一些性质。这种研究路线在机器学习中是非常流行的,不过就近期关于决策树的研究而言,主要还是朝着强化其理论基础的方向前进的。在这本 CART 专著中,理论研讨颇为深入,为读者提供了重要的技术洞见和关键结果的证明。举个例子,作者推导出最大(或最大可能)树的期望错误率的上界是贝叶斯错误率的两倍。作者还讨论了偏差-方差的平衡问题,并且分析了属性的数量如何影响偏差。这本 CART 专著在很大的程度上是以 Richard、Olshen 和 Charles 的早期工作为基础的,其最后三章主要是从理论上研究 CART 同最近邻分类器 NN 之间关系:当样本的大小趋于无穷大时,(1)回归函数的估算将收敛于真正的目标函数;(2)叶节点的风险收敛于贝叶斯规则的风险。换句话说,就是只要具备足够大的样本数据,CART 树就会收敛于真正的目标函数,该目标函数以最小的成本(贝叶斯错误率)实现从预测器(输入)到目标(输出)的映射。但是,要达到这样的性能所需的样本规模远超我们当前可用样本的实际规模。

10.14 CART 之后的相关研究

这部 CART 专著于 1984 年发表后,被大量的科技文献所引用,从中可以看出关于决策树的研究一直很活跃。为了简单起见,这里我们只讨论受到该专著四个作者指导的研究工作。1985 年 Breiman 和 Friedman 提出了替代条件期望 ACE(alternating conditional expectations),这是一种基于回归变量变换的纯数据驱动的方法。这项工作强烈影响了 Hastie 和 Tibshirani 的广义加法模型 GAM(1986 年);Stone(1985 年)为 ACE 的非参数加法回归发展了一套严谨的理论。此后,Friedman 很快就研究出用于样条回归的递归划分法,称为多变量自适应回归样条 MARS(multivariate adaptive regression splines)。我们能找到的最早的 MARS 程序是 1989 年 10 月的 2.5 版,关于 MARS 的第一篇论文是被作为"领头雁"式的文章发表于 1991 年的 *Annals of Statistics* 上。MARS 算法几乎完全基于这

部 CART 专著所提述思想，不过，该算法生成的模型看起来是一种基于递归划分（和选择）预测器的回归过程。后来，Stone 和他人协作又将样条回归方法发展成风险建模 hazard modeling（Kooperberg，Stone and Truong，1995）和多叉回归 polychotomous regression（1997）。

Breiman 则致力于 CART 树方法的精度改进、应用范围拓展和计算速度提升等方面。1992 年，Breiman 第一次在软件中引入了多变量决策树（向量依赖变量），但他没有发表任何关于这个主题的论文。1995 年，Spector 和 Breiman 借助于 C-Linda 的并行编程环境实现了 CART 的并行化。在这项研究中，作者发现“大量数据＋少量处理器”的组合能获得并行收益。1994 年，Breiman 发现了“自举聚合”（bootstrap aggregation）现象：用自举的方法从一个固定的训练数据集中生成大量样本，进而生成大量的 CART 树，聚合这些树就创建出组合预测方法。1998 年 Breiman 将这个组合的思想用于在线学习和开发针对极大规模数据集的分类器。之后，他提出了一种称为随机森林（random forests）的思想，将原来对训练数据的行随机采样推广到对树的每个节点的列随机采样。Breiman 在自己生命的最后几年致力于随机森林的深入研究，他和 Adele Cutler 一起在缺失值的分派、异常的检查、聚类的发现，以及对随机森林算法的输出结果的可视化等诸多方面发展了成体系的新方法，这些成果大都发表在从 2000 年到 2004 年的一系列的论文中和互联网上。

Richard Olshen 研究的重点主要集中于决策树在生物医学方面的应用。例如：他发明了第一个基于树的存活分析方法（Gordon 和 Olshen 1984）；他还成功地将决策树用于图像压缩任务（Cosman 等，1993）；近期，针对极高维数据分析问题，他引入了一种分裂器线性合并方法（复杂疾病的遗传）。

Friedman 从 1999 年开始，提出并发展了随机梯度推举法（stochastic gradient boosting），这种方法是对传统推举法的重大改进，相关成果发表了若干篇学术文章，同时也进行了商业化（TreeNet 软件）。Friedman 的随机梯度推举方法的要点有：生成极小树；在每个训练周期中随机采样训练数据；在每个训练周期中通过更新极小模型实现慢速学习；基于模型残差选择性的抛弃训练数据；允许使用多重目标函数。随机梯度推举法很好地综合了这些特性，因而在许多实际应用上获得了非常好的性能。此后，Friedman 又研究了多种树压缩的新方法，将树组合（tree ensembles）压缩成一种仅含有很少量树的模型，用于正则化回归。这项研究表明将树组合进行压缩作为一种后处理手段可以显著提升决策树在保留数据（holdout data）上的性能。Friedman 沿这条路线继续深入研究，发展了一种将树组合模型转换为规则集合的方法，该方法主要优势是能大幅度地压缩模型，甚至有些情况下还能提高预测的准确性。

可以从 Salford Systems 公司的网站 http://www.salford-systems.com 上找到更多的参考文献和关于 CART 应用的总结。

10.15　可用软件

CART 软件可以从 Salford Systems 公司获取，访问网址 http://www.salford-systems.com；免费的评估版可以直接下载。该软件的可执行文件包括 32 位版和 64 位版，支持的操作系统包括 Windows、Linux 和 UNIX。教授们将 CART 用于学术研究时应选择

学术许可证，该许可证也自动授权其学生免费使用。Jerome 和 Friedman 编写的源码由于已经认证为商业秘密，所以只能得到其编译后的二进制文件。还有一些流行的开源系统（和其他商业化的专有系统）提供了决策树，虽然其原理也是基于 Breiman、Friedman、Olshen 和 Stone 等人的成果，但是当在现实世界的复杂数据集上运行时，你会发现这些系统生成的树与 CART 还是不同的。Salford Systems 公司用 CART 赢得了不少国际上的数据挖掘竞赛，详细的内容可以从该公司的网站上获取。

10.16 习 题

1. (a)对于决策树的新手来说，CART 树中最重要的变量就是根节点分裂器，通常 CART 输出的摘要中不会显示哪个变量是最重要的，请问如何能显示出来？(b)如果先使用 CART 对数据集里的预测变量进行排序并返回具有非零重要性的变量，那么再次运行 CART 能否得到和上次相同的树？(c)如果仅使用 CART 第一次运行获得的分裂器中的变量作为此次运行的预测器，情况又将如何？什么条件是否能确保获得相同的树？

2. CART 树的每个内部节点包含 3 个分裂器：主分裂器、竞争分裂器和替代分裂器。有的树中一个变量可以同时作为竞争器和替代器，只是它们的分裂点不一样，例如，一个变量 x_j 作为竞争者可能用 $x_j \leqslant c$ 来分裂节点，而作为替代者可能用 $x_j \leqslant d$ 来分裂节点。请解释为什么这种情况是可能出现的。

3. CART 常选用 6 种分裂准则里的 Gini 准则和 Twoing 准则来生长树。请解释一下为何 Twoing 准则的结果即便具有较低的准确度还是能受到人们的青睐？

4. 在同一个数据集上生长两个二分类的 CART 树，第一个使用 Gini 分裂准则，第二个使用类概率准则，请问哪一个可能含有更多的节点？两个树是否具有相同的准确度？小的那个树是否包含于大的那个树里？这两个树有何差别？

5. 假设有一个二分类(0/1 分类)数据集，其中 80%的记录属于类别 0。先用默认的“先验相等”(PRIORS EQUAL)来生长 CART 树，然后(记权重变量为 w)对类别 0 取 w=1，类别 1 取 w=4，生成的树结果会有什么变化？

6. 如果数据集很大包含了数以万计的记录，生长的大 CART 树与最优 CART 树相比，其准确度仅有微小下降。也就是说对于大数据集而言，大 CART 树的过拟合现象很轻微。请解释这是为什么？

7. 一个 CART 模型不仅仅是一棵树，还可以看成是一个树的集合，而且其中的每棵树都具有自己的性能特征（准确度、ROC 曲线的下部面积）。为什么 CART 发明人建议：最佳的树不一定是准确度最高的树，而应在最高准确度的某个容忍区间内从树序列中取较小尺寸的树？进一步，又如何来计算这个容忍区间呢？

8. 不同的错误应该与不同的代价相关联，为了进行代价敏感学习，CART 发明人通过调整先验来反映代价，这种方法本质上是一种数据加权。什么情况下先验的调整能很好地反映代价，而什么情况下仅能逼近代价？对称 Gini 分裂准则是如何反映错分代价的？

9. CART 发明人在选择最优决策树时选择了“生长-剪枝”策略而不是简单的“停止准则”方法。请说明一下，异或型(XOR-type)问题可以让任何“停止准则”失效，不论使用哪种分裂准则。

10. 如果训练集的数据是完整的(任何预测器都没有缺失值)，由此得到的 CART 树是如何确保能够处理具有缺失值的新数据的？

参 考 文 献

[1] Bloch, D. A., Olshen, R. A., andWalker M. G. (2002) Risk estimation for classification trees. *Journal of Computational & Graphical Statistics*, 11, 263-288.

[2] Breiman, L. (1995) Current research in the mathematics of generalization. *Proceedings of the Santa Fe Institute CNLSWorkshop on Formal Approaches to Supervised Learning*. David Wolpert, Ed. Addison-Wesley, 361-368.

[3] Breiman, L. (1998) *Pasting Small Votes for Classification in Large Databases and On-Line*. Statistics Department, University of California, Berkeley.

[4] Breiman, L., and Friedman, J. H. (1985) Estimating optimal transformations for multiple regression and correlation. *Journal of the American Statistical Association*, 80, 580-598.

[5] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984) *Classification and Regression Trees*, Wadsworth, Belmont, CA. Republished by CRC Press.

[6] Breiman, L. and Stone, J. (1978) Parsimonious Binary Classification Trees, Technical Report, Technology Services Corp., Los Angeles.

[7] Cosman, P. C., Tseng, C., Gray, R. M., Olshen, R. A., et al. (1993) Tree-structured vector quantization of CT chest scans: Image quality and diagnostic accuracy. *IEEE Transactions on Medical Imaging*, 12, 727-739.

[8] Cover, T. and Hart, P. (1967) Nearest neighbor pattern classification, *IEEE Trans Information Theory 13*, page(s): 21-27.

[9] Domingos, P. (1999) MetaCost: A general method for making classifiers costsensitive. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pp. 155-164.

[10] Doyle, P. (1973) The use of automatic interaction detector and similar search procedures. *Operational Research Quarterly*, 24, 465-467.

[11] Einhorn, H. (1972) Alchemy in the behavioral sciences. *Public Opinion Quarterly*, 36, 367-378.

[12] Friedman, J. H. (1977) A recursive partitioning decision rule for nonparametric classification. *IEEE Trans. Computers*, C-26, 404. Also available as Stanford Linear Accelerator Center Rep. SLAC-PUB-1373 (Rev. 1975).

[13] Friedman, J. H. (1999) *Stochastic Gradient Boosting*. Statistics Department, Stanford University.

[14] Friedman, J. H., Bentley, J. L., and Finkel, R. A. (1977) An algorithm for finding best matches in logarithmic time. *ACM Trans. Math. Software*, 3, 209. Also available as Stanford Linear Accelerator Center Rep. SIX-PUB-1549, Feb. 1975.

[15] Friedman, J. H., Kohavi, R., and Yun, Y. (1996) Lazy decision trees. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 717-724, AAAI Press/MIT Press, San Francisco, CA.

[16] Gordon, L., and Olshen, R. A. (1985) Tree-structured survival analysis (with discussion). *Cancer Treatment Reports*, 69, 1065-1068.

[17] Gordon, L., and Olshen, R. A. (1984) Almost surely consistent nonparametric regression from

recursive partitioning schemes. *Journal of Multivariate Analysis*, 15, 147-163.

[18] Hastie and Tibshirani's Generalized Additive Models. (1986) *Statistical Science*. 1, 297-318.

[19] Huang, J., Lin, A., Narasimhan, B., et al. (2004) Tree-structured supervised learning and the genetics of hypertension. *Proc. Natl. Acad. Sci.*, July 20, 101(29), 10529-10534.

[20] Kooperberg, C., Bose, S., and Stone, C. J. (1997) Polychotomous regression. *Journal of the American Statistical Association*, 92, 117-127.

[21] Kooperberg, C., Stone, C. J., and Truong, Y. K. (1995) Hazard regression. *Journal of the American Statistical Association*, 90, 78-94.

[22] Messenger, R. C., and Mandell, M. L. (1972) A model search technique for predictive nominal scale multivariate analysis. *Journal of the American Statistical Association*, 67, 768-772.

[23] Morgan, J. N., and Sonquist, J. A. (1963) Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, 58, 415-435.

[24] Provost, F., and Domingos, P. (2002) Tree induction for probability-based ranking. *Machine Learning*, 52, 199-215.

[25] Quinlan, R. (1989) Unknown attribute values in induction. In *Proceedings of the Sixth International Workshop on Machine Learning*, pp. 164-168.

[26] Stone, C. J. (1977) Consistent nonparametric regression (with discussion). *Annals of Statistics*, 5, 595-645.

[27] Stone, C. (1985) Additive regression and other non-parametric models, *Annal. Statist.*, 13, 689-705.

[28] Ting, K. M. (2002) An instance-weighting method to induce cost-sensitive trees. *IEEE Trans. Knowledge and Data Engineering*, 14, 659-665.